2014

# A Measure of Vision Distance for Optimization of Camera Networks

Jose Alarcon
*University of Windsor*

# A Measure of Vision Distance for Optimization of Camera Networks

by

Jose Alarcon

A Dissertation
Submitted to the Faculty of Graduate Studies
through the Department of Electrical and Computer Engineering
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
at the University of Windsor

Windsor, Ontario, Canada
2014

# A Measure of Vision Distance for Optimization of Camera Networks

by

Jose Alarcon

Approved by:

---

Dr. F. Zhang, External Examiner
School of Electrical and Computer Engineering, Georgia Institute of Technology

---

Dr. G. Zhang
Department of Mechanical, Automotive and Materials Engineering

---

Dr. M. Ahmadi
Department of Electrical and Computer Engineering

---

Dr. B. Boufama
Department of Electrical and Computer Engineering

---

Dr. X. Chen, Advisor
Department of Electrical and Computer Engineering

September 12, 2014

# Declaration of Previous Publication

This dissertation includes two original papers that have been previously published/submitted for publication in peer reviewed journals, as follows:

| Thesis Chapter | Citation | Status |
|---|---|---|
| Section 2.2 | A. Mavrinac, X. Chen, and J.L. Alarcon-Herrera, "Semi-Automatic Model-Based View Planning for Active Triangulation 3D Inspection Systems ," to appear in *IEEE/ASME Transactions on Mechatronics*, DOI:10.1109/TMECH.2014.2318729, 2014. | Accepted |
| Chapter 3 | J.L. Alarcon-Herrera and X. Chen, "Graph-Based Deployment of Visual Sensor Networks and Optics Optimization," submitted to *IEEE/ASME Transactions on Mechatronics*, manuscript no. TMECH-07-2014-3852. | Submitted |
| Chapter 4, Chapter 5, Chapter 6 | J.L. Alarcon-Herrera, X. Chen, and X. Zhang, "A Tensor-Based Vision Distance for Optimization of Multi-Camera Systems," submitted to *IEEE Transactions on Pattern Analysis and Machine Intelligence* | to be Submitted |

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my dissertation. I certify that the above material describes work completed during my registration as graduate student at the University of Windsor.

I declare that, to the best of my knowledge, my dissertation does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my dissertation, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my dissertation.

I declare that this is a true copy of my dissertation, including any final revisions, as approved by my dissertation committee and the Graduate Studies office, and that this dissertation has not been submitted for a higher degree to any other University or Institution.

# Abstract

This dissertation proposes a solution to the problem of multi-camera deployment for optimization of visual coverage and image quality. Image quality and coverage are, by nature, difficult to quantify objectively. However, the chief difficulty is that, in the most general case, image quality and coverage are functions of many parameters, thus making any model of the vision system inherently complex. Additionally, these parameters are members of metric spaces that are not compatible amongst themselves under any known operators.

This dissertation borrows the idea of transforming the mathematical definitions that describe the vision system into geometric constraints, and sets out to construct a geometrical model of the vision system. The vision system can be divided into two different concepts: the camera and the task. Whereas the camera has a set of parameters that describe it, the task also has a set of task parameters that quantify the visual requirements. The definition of the proposed geometric model involves the construction of a tensor; a mathematical construct of high dimensionality which enables a representation of the camera or the task. The tensor-based representation of these concepts is a powerful tool because it brings a large tool set from various disciplines such as differential geometry.

The contributions of this dissertation are twofold. Firstly, a new distance function that effectively measures the distance between two visual entities is presented based on the geometrical model of the vision system. A visual entity may be a camera or a task. This distance is termed the Vision Distance and it measures the closeness to the optimal state for the configuration between the camera and the task. Lastly, a deployment method for multi-camera networks based on convex optimization is presented. Using second-order cone programming, this work shows how to optimize the position and orientation of a camera for maximum coverage of a task.

This dissertation substantiates all of these claims. The vision distance is validated and compared to an existing model of visual coverage. Additionally, simulations, experiments, and comparisons show the efficacy of the proposed deployment method.

*To my Mexican family,*
*and to my Serbian family.*

# Acknowledgements

First I would like to thank Dr. Xiang Chen. As my advisor and mentor, he has provided invaluable guidance during my studies over the past four years. His vision and high standards have made a profound change in my way of thinking. But most important, I thank Dr. Xiang Chen because he showed me the forest when a tree was all I could see. I also thank the other members of my committee, Dr. Majid Ahmadi, Dr. Boubakeur Boufama, and Dr. Guoqing Zhang. They provided valuable feedback toward my research, and generously spent their time reviewing my work.

I owe much gratitude to my fellow students Aaron Mavrinac, Davor Srsen, and Xuebo Zhang. The work in this dissertation owes much to the countless and productive discussions we had over the years. In particular, Aaron's dedication and contributions to society and the environment have always inspired me.

I am indebted to the family and friends who supported me in more ways than I can recall. My parents, Jose and Olivia, who supported me and encouraged me to pursue graduate studies even when faced by adversity. Ilinka Srsen, whom I think of as a mother, welcomed me into her home when I found myself alone in a foreign land. In particular, I am grateful to my cousin, Daniel, who gave me invaluable advise and who supported me more than anyone else in one of the most important moments of my life.

My deepest gratitude goes to my aunt and uncle. My aunt, Teresa, is a hard working and highly accomplished researcher. From an early age she has always been a source of inspiration for pursuing graduate studies. My uncle, Ignacio, is the wisest man I know. I have been lucky enough to be able to learn from him. His teachings have helped me live a life with integrity and honesty.

Finally, I would like to acknowledge the generous financial contributions of the National Council of Science and Technology of Mexico, the Natural Sciences and Engineering Research Council of Canada, and the University of Windsor towards this research.

# Contents

# List of Tables

# List of Figures

# List of Algorithms

If I have seen further it is by standing on the shoulders of giants.

---

Isaac Newton (1643–1727), *Letter to Robert Hooke*

# Introduction

# CHAPTER 1

# Introduction

Perseverance is the only indicator of success.

Daniel Martin-Alarcon (1988–)

## 1.1 Optimization of Camera Networks

Optimization of camera networks is a broad topic, and often it implies that some metric of performance, whether quantitative or qualitative, can be maximized or minimized by means that may be direct or indirect. This dissertation attempts to optimize a custom metric of performance, indirectly, by means of optimal camera deployment. This implies that the relative position and orientation of the camera, with respect to the scene, bears some relationship with the performance of the vision system. Indeed, the performance of the vision system is a function of the many variables that parametrize the vision system; the position and orientation among those.

Measuring the performance of the vision system is by itself a challenging task, and the parametrization of the vision system is directly related to the quantification of visual coverage and image quality. Often, the best way to bring all these concepts together is to define a model for the vision system. Such model may consider two main components: the camera and the task, and it may define performance as a bounded scalar. The camera may be modeled separately, and described by a set of internal and external parameters. On the other hand, the task represents the objective of the vision system. Based on the model of the camera, the task is often represented as a set of three-dimensional points in space, which may sometimes carry a topological structure. Additionally, it is common to append to the task a set of parameters, which quantify the task's own visual requirements. Mavrinac and Chen [1] offer an excellent survey on different models of the vision system.

The task parameters fully describe the requirements such that the task can be imaged with sufficient quality in any arbitrary camera. The early works of Shi and Tomasi [2] and Tarabanis et al. [3] show some of the rationale by which the task parameters are set so that they can quantify concepts such as image quality. More recently, Mavrinac [4] proposed a task model with eight different task parameters. The task as a set of three-dimensional points is a useful concept because it can be used to model areas or volumes, with an arbitrary degree of precision, which themselves model parts of

the scene that are of interest or the objective to be covered by the vision system. Additionally, when the task carries a topological structure, such as a neighbourhood system, the task may be defined as a triangular mesh (e.g a CAD model), which can be used to model physical objects in the scene.

The model of the vision system may be formulated with varying levels of complexity. This depends on the number of parameters included in the model, and the linearity of the functions used to relate them to the overall performance of the vision system. In general, vision systems that use simple yet inaccurate models are easy to optimize. Whereas systems with more complex models are more difficult to optimize and may limit the solution to a suboptimal one.

## 1.2 Motivations

Camera networks have applications in many areas of science and technology. In environmental science, camera networks play a crucial role by enabling the acquisition of data that would otherwise be hard to obtain. For example, the extreme ice survey [5] utilizes cameras to present a visual record of melting glaciers. Environmental science and robotics find an intersection in the work of Casbeer et al. [6] and Merino et al. [7], where the authors use camera networks to monitor forests and detect forest fires. In industrial settings, camera networks are used extensively in the area of robotics for the development of autonomous vehicles [8, 9, 10]. On the other hand, there are several areas of robotics, such as consensus-base motion control [11, 12, 13, 14], that could receive much improvement from the use of vision systems. In the manufacturing industry camera networks are particularly good at performing high accuracy and non-contact metrology for inspection [15].

In applications of visual surveillance and security the work of Qureshi and Terzopoulos [16] shows the means for capturing high-resolution images of pedestrians and other targets in an automatic fashion. Hu et al. [17] and Lepetit and Pascal [18] presented excellent surveys on visual surveillance. Surveillance, as well as many other applications, can be used to make a strong case for automation. Specific applications such as activity recognition, and target identification becomes a difficult set of tasks for human operators as the covered area increases, hence the need for automatic or semi-automatic solutions which scale well while performing with sufficient quality. Additionally, surveillance systems controlled by human operators also have ethical implications; as reported by the Information and Privacy Commissioner of Ontario [19], the ability of operators to adjust the cameras should be restricted, to prevent the surveillance system from imaging spaces not intended to be covered, strengthening the need for automatic methods of surveillance.

### 1.2.1 Camera Deployment

The problem of optimal camera deployment is defined as the computation of the position and orientation of a camera or cameras, such that the performance metric of the vision system be maximized with respect to a given task. Optimizing the deployment of a single camera is a constrained version of the multi-camera deployment problem. On the other hand, the deployment of multiple cameras carries an additional difficulty, which is that the cost must be minimized (i.e. the solution should have as few cameras as possible). The tradeoff is clear: more cameras increase the area coverage; however, less cameras reduce the cost of the solution. Furthermore, a camera's performance may vary depending on the angle of view (i.e. the orientation). In other words, the different effects on the

performance induced by the position, orientation, and the number of cameras in a camera network are not independent from one another, thus making the problem hard to solve.

Optimal camera deployment is a desirable objective because it improves the performance of the vision system, according to the definition of the performance metric. Additionally, multi-camera deployment has the potential to automate the implementation of surveillance systems and the control of mobile robotics in several other applications. The deployment of cameras is often regarded as an offline problem, since the solution is computed prior to the implementation of the camera network.

### 1.2.2 Camera Reconfiguration

The problem of camera reconfiguration is similar to the problem of camera deployment in that some parameters must be determined, usually orientation, with the objective of achieving some goal. Examples are covering certain areas of the scene or maintaining a target or targets in the field of view of one or more cameras at all times. The problem of camera reconfiguration is a dynamic one. Camera reconfiguration targets applications that have goals or objectives that are varying in time. For example, when tracking objects in the scene, objects are often moving as it is the case with surveillance applications. Alarcon and Chen. [20] presented a method for reconfiguration of pan-tilt-zoom cameras. Also, in [21] the authors present an application of the problem of camera reconfiguration in the setting of unmanned aerial vehicles.

In contrast, the problem of camera reconfiguration assumes that some parameters are fixed or have been determined a priori, usually position and the number of cameras. In most cases, the control over the orientation is also limited, for practical reasons. Although the number of cameras is fixed in this case, the problem may incorporate the activation or deactivation of any number of cameras for the purpose of resource allocation; thus bringing back some form of the cost minimization problem. An example of the resource allocation is found in Mavrinac and Chen [22].

The problem of camera reconfiguration is not limited to the orientation or the number of cameras. Since pan-tilt-zoom cameras allow the variation of the optical parameters over time, the optimization of the optics is also a commonly pursued goal. The difficulty in this case comes from the non-linear behaviour of most lenses. Alarcon et al. [23] propose a geometric approach that finds a tradeoff between the several variants in the configuration of the optical parameters.

### 1.2.3 Camera Selection

Camera selection deals with the selection of a camera that can provide the best view of the scene at a given instant in time. In most cases, but not all, the position, orientation, and the number of cameras are fixed. The major difficulty is to provide a smooth transition between cameras when the system changes the view. Mavrinac et al. [24] present an example of this type of work. Applications of this work are surveillance and coverage of sport events, the later being the most common.

### 1.2.4 Next Best view

The next best view problem is most easily understood as an online version of the multi-camera deployment problem. In this case the objectives are to minimize the number of views needed, and to determine the optimal position and orientation of every view in a sequence. In addition, a single

camera is commonly used and its parameters are changed in sequence in order to cover all necessary views. The work of Pito [25] and Scott [15] are examples of the next best view problem. Some applications include highly-accurate 3D scanning or reconstruction.

## 1.3 Proposition

This dissertation approaches the problem of optimization of camera networks via optimal deployment of multiple cameras.

Based on some models of the vision system found in the literature, this dissertation relies on the idea of transforming the various camera parameters and task parameters into geometric constraints. Existing work has been done using a similar idea [26, 27]. Some models use the set of geometric constraints to formulate a model based on set theory [4], allowing a meaningful integration of the various components that contribute to the performance of the vision system. Other approaches include the construction of a model based on graph theory [28, 29, 30]. This last approach enables the representation of aspects of the camera network that may not be directly related to visual performance, but that allow the implementation of distributed architectures. This dissertation differs from previous approaches in that the geometric constraints are used to formulate a geometric model of the vision system.

The geometric model of the vision system is based on the construction of a tensor, the tensor is a collection of orthogonal vectors that, together, carry information about the orientation of the camera and the size and shape of the viewing frustum. Similarly, a tensor is constructed to represent a triangle, which is later defined as the atomic unit that represents a task. Using the camera and triangle tensors as operands, additionally accounting for position, this dissertation defines a vision distance that effectively measures the distance between a camera and a task. The vision distance combines the Euclidean distance with a scalar that is the semi-direct result of a distance on the space of rotations; thus considering the distance between the orientations as well. In this framework a task is considered fully covered –with acceptable resolution, focus, and view angle– when the distance is 0.

Based on the vision distance, an optimization approach is formulated for multi-camera deployment. The vision distance is minimized by separately minimizing the Euclidean distance and the distance between the orientations. The optimization is divided into three parts. The first part of the problem is formulated, using convex programming, as the search of the position of the camera that minimizes the Euclidean distance to a set of triangles that form a task. The second part is formulated as the search of the orientation of the camera that minimizes the orientation distance to a set of triangles in the task. Finally the third part is the deployment of multiple cameras using the individual deployment process of the first and second parts.

One of the major difficulties encountered in this work is the minimization of the orientation distance. Due to the higher dimensions needed to represent an orientation in the conventional sense, standard convex programming is not possible. This dissertation shows the means by which the orientation problem can be transformed into a space of lower dimension. Thus reducing the problem to the Euclidean domain, and allowing standard convex programming to be used to find a solution.

## 1.4 Thesis Outline

This dissertation begins in Part I with a survey, in Section 2.1, of the state of the art in optimization of camera networks through camera deployment. Section 2.1 also presents a few examples of work relating to the camera reconfiguration problem as well as the camera selection and next best view problems. Section 2.2 briefly introduces the necessary parts of the coverage model from wich the tensor framework is based. Chapter 3 presents previous work that is directly related to the deployment of multiple cameras. This work is used in this dissertation later in Chapter 6 and Chapter 7.

The first major contribution of this dissertation is found in Part II. The tensor framework is presented in Chapter 4. The camera and triangle tensors are defined in Section 4.1 and Section 4.2, respectively. Section 4.3 defines some operations on tensors that are used throughout this dissertation. The vision distance is defined and presented in Chapter 5. The Frobenius norm is presented as a norm for elements in the space of rotations and a distance for such elements is induced from this norm in Section 5.1. The vision distance is formally presented in Section 5.2, and proof is provided about some of its properties.

Part III presents the second major contribution of this dissertation. Firstly, a method for deployment of camera networks is offered in Chapter 6. The optimization of the position and orientation of individual cameras are presented in Section 6.2 and Section 6.3, respectively. Section 6.4 takes advantage of the work previously presented in Chapter 3 and extends it by finalizing the deployment method with the tools presented in Section 6.2 and Section 6.3. Finally, Chapter 7 presents a method for deployment of camera networks for surveillance applications, which is itself an application case of the optimization framework presented in this dissertation.

This dissertation presents a collection of experiments, simulations, and comparisons, with existing work, that validate the claims made throughout it. The experimental validation of the work presented in Chapter 3 can be found in Section 3.6. The validation of the accuracy and usefulness of the vision distance may be found in Section 5.3. Section 6.5 presents the main results of the optimization scheme, and compares its performance with the performance of the method presented in Chapter 3. Finally, the optimization framework of this dissertation is compared to an existing method for deployment of camera networks in Section 7.4.

Chapter 8 offers some concluding remarks, which summarize the contributions presented in this dissertation. Additionally, some potential directions for future work are outlined.

The appendices cover some of the mathematical background used in this dissertation. Appendix A reviews several mathematical concepts and conventions used throughout the dissertation. Appendix B lists the specifications for the equipment used in the various experiments.

# Part I

# Previous Work

# CHAPTER 2

# State of the Art

> Books permit us to voyage through time, to tap the wisdom of our ancestors.
>
> Carl Sagan (1934–1996), *Cosmos*

## 2.1 Overview

The literature is rich with examples of work addressing the problems described in Section 1.2. A few of these examples are presented next. Additionally, some relevant remarks are made according to the way the authors formulate the optimization problem and the tools used to approach it.

### 2.1.1 Camera Deployment

#### Set Covering

The various tools by which optimization is performed, in the context of camera deployment, range widely from convex programming to brute force algorithms. Within the range, set covering is commonly found as several examples show a formulation of the original problem accompanied by a reduction to the set covering problem. In [26] Cowan and Kovesi offer a method for automatic sensor placement based on the task requirements. The approach converts the task requirements into geometric constraints. The method provides a unified scheme for sensor planning. Tarabanis et al. [27, 31] approach the problem of computing viewpoints that satisfy some task requirements and avoid occlusion. The approach is similar to that of Cowan and Kovesi [26], in that the task requirements are transformed into geometric constraints on the sensor's location.

Angella et al. [32] worked on the optimal deployment of cameras in a three-dimensional scene by transforming all the visual constraints into distances, and reducing the problem to the set covering problem. In the area of industrial inspections Scott [15] proposes a solution to the viewpoint selection problem by first defining a solution space and generating a matrix that measures visual coverage according to some custom metric. Then by reducing the problem to the set covering problem, the solution is found by a greedy search method that operates over the entries of the matrix, where the

rows represent points in the scene and the columns represent viewpoints. The solution is then to cover as many rows with as few columns as possible.

### Genetic Algorithms

Genetic algorithms are well suited to approach the problem of multi-camera deployment. In this framework cameras represent individual agents or particles. Commonly, a fitness function is defined based on the performance function of the vision system, and the gene pool is defined as a set of configurations of camera parameters. This approach is usually implemented in an iterative fashion, and the genes are allowed to reproduce and mutate, according to some custom rule. Eventually the algorithm converses to a feasible solution. In [33] Olague and Mohr apply a genetic algorithm to the problem of camera placement, by defining a fitness function based ton the visual requirements. Zhong at al. [34] present a mixture of multi-agent and genetic algorithms to find a solution to the camera placement problem that is globally optimal. The approach shows good results at a low computational cost; however, the method is highly sensitive to perturbations to the relative positions of the cameras. In [35] Malik and Bajcsy set out to find optimal camera placements for stereo applications. The authors use a genetic algorithm to find an initial solution. The initial solution is then refined using a gradient descent algorithm.

Wang et al. [36] propose a method to find camera placements that maximize the observation based on a custom sensing model. In this case, a multi-agent genetic algorithm is used to find a solution. In [37] Jiang et al. propose a method to find camera locations that maximize some custom metric of weighted coverage, that gives preference to a particular objective as defined by the user. The authors make use of a genetic algorithm to find a solution to the problem. More recently in [38] Reddy and Conci have solved automatic camera positioning using a particle swarm optimization algorithm, which operates in two dimensions. The fitness function in this case considers resolution, quality of view, and light intensity. Another example of an application of particle swarm optimization can be found in [39], where Mavrinac et al. solve the problem of sensor deployment for 3D active cameras. The authors formulated a fitness function based on a coverage model of the vision system. The authors optimized the position and orientation of the camera with respect to the laser plane used for active triangulation.

### General Heuristics

General heuristic and meta-heuristic methods are commonly used in the area of camera deployment. In some cases these methods are the main tool used for optimization. However, in some cases heuristics are used as an alternative due to the complexity or the computational cost of some optimization formulations.

Bodor et al. [40, 41] have proposed a method for camera placement that pays special attention to the task requirements, and uses statistics of the motion paths in the scene to optimize the solution. Ultimately, the hill climbing technique is used to compute the solution. In [42] González-Banos and Latombe have presented a method to find the minimum set of guardians inside a polygonal space from which the entire boundary is visible. In this work the authors offer a randomized version of the well-known art gallery problem [43, 44, 45] tailored for sensor planning. The authors used a greedy algorithm to compute an approximate solution to the problem. In [46] Erdem and Sclaroff tackled

the problem of viewpoint selection and cost minimization for specific task requirements. The authors propose a reduction of the problem to the set covering problem by mapping the field of view to feasible regions in two dimensions, and by mapping the task-based constraints of the system to area coverage constraints for camera layout in floor plans. In [47, 48] Mittal and Davis approached the problem of computing camera positions for coverage of dynamic scenes. The authors proposed a probabilistic model of the scene and use simulated annealing to minimize some custom cost function.

Hörster and Lienhart [49] also approached the optimal placement of cameras in a two-dimensional scene by showing a way to compute an optimal solution using binary integer programming. However, the solution was found using a greedy algorithm due to the high cost of computing the original solution. In [50] Chen and Davis developed a quality metric that accounts for dynamic feature occlusion. The authors develop a function that expresses the probability that at least some lower bound of coverage can be satisfied. The study of worst case scenario is used to avoid unfeasible computations. Kansal et al. [51] proposed a distributed algorithm with network interactions and global utility maximization through the optimization of the local utilities of the cameras. The local utilities are informed by a model of visual coverage. Zhao et al. [52] presented a method for optimal camera placement in visual tagging applications. The authors formulated the problem using binary integer programming; however, a greedy algorithm is used instead due to the unfeasibility of computing the original solution. Krause et al. [53] presented a method to evaluate and select robust sensor placement with special consideration for communication efficiency between the sensors. The authors use a probabilistic model of coverage, which in turn was used by an algorithm that evaluates and finds optimal sensor locations. The algorithm used a greedy search as well as the expected quality and the communication costs to find good sensor locations.

Although few, one can still find examples of brute force programming for the solution of some form of the camera deployment problem. Ram et al. [54] proposed a design method for placement of sensors in surveillance applications. The authors approached the problems of placement, cost reduction, and failure behaviour. In this work the authors employed a generate-and-test approach in order to find feasible solutions. Mackay et al. [55] provided an implementation strategy for a method that aims to find camera configurations in real time. The authors used a visibility model and a Kalman filter to estimate future target locations, and the camera configurations are found using a generate-and-test approach.

### 2.1.2 Camera Reconfiguration

In most cases solutions to the camera reconfiguration problem are applied to the reconfiguration of pan-tilt-zoom camera networks. In [56] Fiore et al. set out to optimize the external parameters of the camera such that a network of cameras can adapt to a dynamic scene. The authors defined an absorbability metric that is a function of the pose of the camera and the focal length. The authors used a brute force computation to maximize through all cameras using this observability metric. Piciarelli et al. [57, 58, 59] presented a method for PTZ camera reconfiguration that represents the scene as a discrete space of 3D points and a relevance map that weighs the importance of the points. The scene is modeled by a mixture of Gaussians, the camera's field of view is modeled by an ellipse with a pan-tilt-zoom configuration, and the problem is reduced to fitting ellipses to a data set using the expectation-maximization algorithm.

Qureshi and Terzopoulos [16, 60, 61] and Starzyk and Qureshi [62] develop a system to provide strategies for camera assignment and camera handoff that becomes computationally efficient over time. The authors use a combination of static and active cameras. Then, using a reasoning module, the necessary actions are computed, generalized, and stored in a production system. When the system encounters a familiar situation it avoids computation by using the stored actions instead.

### 2.1.3 Camera Selection and the Next Best View Problem

Examples of camera selection and the next best view problems are briefly reviewed. As an example of the former the reader may refer to Chow et al. [63, 64, 65] who presented a solution to the problem of scheduling sensors in order to minimize transmission costs and reduce redundant data. The authors payed special attention to the optimization of the angle of view. In this work the authors used graph theory to transform the problem into that of finding the shortest path. An example of the later may be found in the work of Pito [25] where the author presents a solution to the next best view problem. The method targeted automated surface acquisition for 3D scanning. The solution involved keeping a model of the scanned areas and computing the areas that need to be scanned. The viewpoints are generated from a model of the unscanned areas and the overall model is updated constantly.

### 2.1.4 Applications of Differential Geometry

In recent years the field of differential geometry has gained attention in the computer vision community. Many concepts and abstract ideas in computer vision are too complex, and thus require the employment of more elaborate tools. Tensors are a good fit to meet this need. The work of Yang et al. [66] and Sivalingam et al. [67] show excellent examples of this approach. The work of Tung and Matsuyama [68] shows how the idea of differential geometry can be used to perform surface alignments, where surfaces are objects of high dimensionality.

Tron et al. [69] offered a method for pose averaging in a distributed fashion. The authors presented reasons why regular average does not translate into a distributed framework, and present a way to overcome some of these difficulties in the tangent space of the space of rotations. In [70] Soto et al. optimized the pose estimation of multiple targets from multiple cameras using the distributed collaboration between all the relevant cameras. A distributed consensus protocol is used to fuse the pose estimation of the targets from the measurement of multiple cameras and update a linear model of the target's dynamics. The authors used spatial adaptive play, from game theory, to assign cameras to targets and reconfigure the camera netwrok. Song et al. [71] proposed a method to optimize pose tracking and activity recognition of pedestrians in a distributed manner. The pose estimation is optimized by "averaging" the estimates of a target using all available estimates of the target. The distributed nature of the consensus protocol allows it to cope with disturbances to the instantaneous estimations over large periods of time.

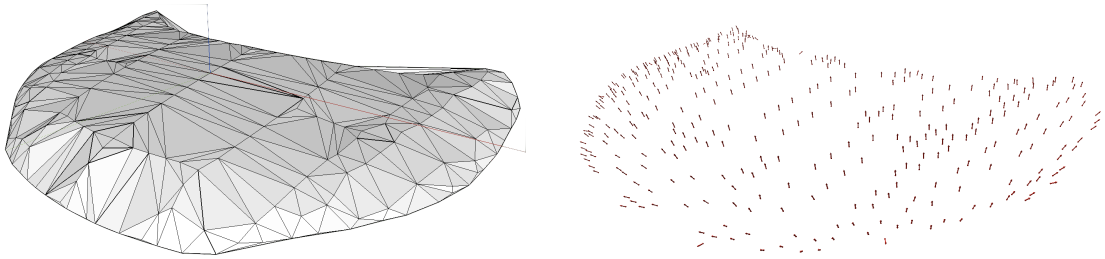
In [69, 70, 71] the authors performed pose averaging by separately obtaining the average of the translational part, in Euclidean space, and the average of the rotational part, in the group of rotations. In fact, the authors first defined a distance metric for rotation matrices and refer to the later as average in the space of rotations. Thus making the group of rotations a metric space. Other examples of this type of work may be found in the area of medical imaging with applications to magnetic resonance

imaging [72, 73, 74, 75]. In this work, a tensor is constructed from the diffusive properties of water molecules in brain tissue. This tensor is called the diffusion tensor, and a distance metric is defined locally for neighbouring tensors in the data set. This distance metric is used to derive measures of structural similarity, and fiber-tract organization.

Although there is some recent work on the applications of differential geometry to solve some interesting problems in computer vision. There is still little evidence that it is being applied to solve the problem of optimal camera deployment. This dissertation relies heavily on the idea of using a tensor construct to represent some important aspects of the vision system. Additionally, by defining a distance metric for such tensors, an optimization scheme is proposed for optimal camera deployment.

## 2.2 Modelling the Vision System[1]

The coverage model of a vision system models multiple cameras, the environment, and the task. The camera system is approximated using the pinhole camera model [76], which accounts for the sensor's properties; additionally the coverage model takes into consideration the lens' properties [77]. The environment model is a set of three-dimensional surfaces and it accounts for deterministic occlusion in the scene. The task is modeled using a set of directional points, which are three-dimensional points with a direction component, and a set of task parameters. The directional points model the target to an arbitrary degree of precision, and the task parameters model the visual requirements of the task.



(a) Triangular Mesh (b) Directional Points

**Figure 2.1**: Task Model – Two different implementations of the task model.

The coverage function $C(v, p)$ is bounded to the range $[0, 1]$ and it represents the grade of coverage at point $p$ as seen from viewpoint $v$. In order to represent the grade of relevance of each point to the task, the relevance function $R(p)$ is used and it is also bounded to the range $[0, 1]$. Finally the coverage performance of the vision system with respect to the task is given by

$$F(C, R) = \frac{\sum_{p \in \langle R \rangle} C(p)R(p)}{\sum_{p \in \langle R \rangle} R(p)} \tag{2.1}$$

in this case $C(p)$ represents the coverage at point $p$ in the multi-camera case. $\langle R \rangle$ is a finite, discrete task point set induced by $R$. $p$ is a directional point in the scene.

[1]This dissertation incorporates the outcome of a joint research which Aaron Mavrinac has undertaken in collaboration with myself under the supervision of professor Xiang Chen. The material presented in this section is included for completeness and to achieve a self contained dissertation.

### Task Model

In this section the task is modeled as a set of points with a weak topological structure. In the equivalent triangular mesh representation, a directional point represents the centre of a triangle and its direction component represents the normal to the plane of the triangle. For the remainder of the chapter the task is represented as a set of directional points; however, everywhere else in this dissertation the task will be represented as a triangular mesh unless otherwise stated. Figure 2.1 shows a comparison between the two froms.

The visual requirements of the task are quantified by a set of parameters listed in Table 2.1[2]. The subscript $i$ stands for "ideal" and $a$ stands for "acceptable"; the latter should be set to the limit at which the task produces no useful information, and the former should be set to the point beyond which the task performance does not increase. The task requirements are defined as follows:

**Table 2.1**: Task Parameters

| Parameter | Description |
|---|---|
| $R_i$, $R_a$ | minimum ideal/acceptable resolution (l/px) |
| $c_i$, $c_a$ | maximum ideal/acceptable blur circle (px) |
| $\zeta_i$, $\zeta_a$ | maximum ideal/acceptable view angle (a) |

1. *Resolution.* Resolution measures the number of units of length that are represented by a pixel in an image sensor. The resolution parameter represents the need for accuracy in some applications (see Figure 2.2).

2. *Blur circle.* The blur circle requirement captures the need for sharpness in the image by setting a threshold on its diameter (see Figure 2.3).

3. *View Angle.* The view angle parameter represents the angle at which objects in the scene become self-occluded from the camera (see Figure 2.4).

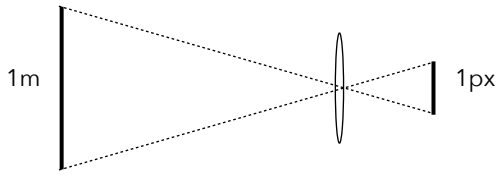



**Figure 2.2**: A visualization of the resolution parameter

### Coverage Function

The coverage model considers four components: resolution, focus, view angle, and visibility. These components can be transformed into geometric constraints as depicted in Figure 2.5. The geometric

[2]Task parameters are defined in units of length (l), pixel units (px), and/or angle units (a).

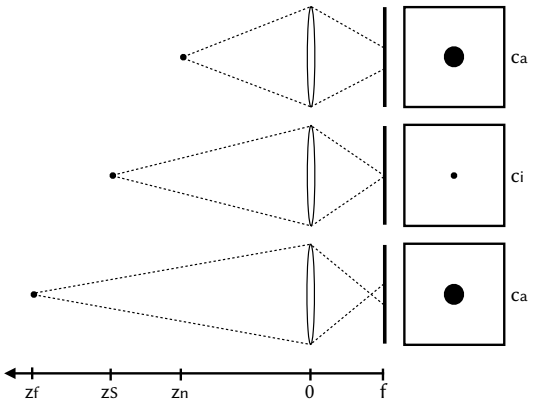



**Figure 2.3:** A visualization of the focus parameter

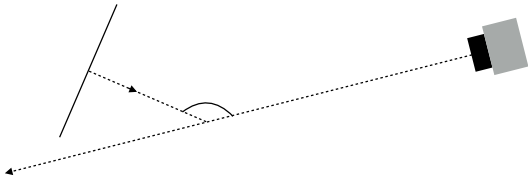


**Figure 2.4:** A visualization of the view angle parameter

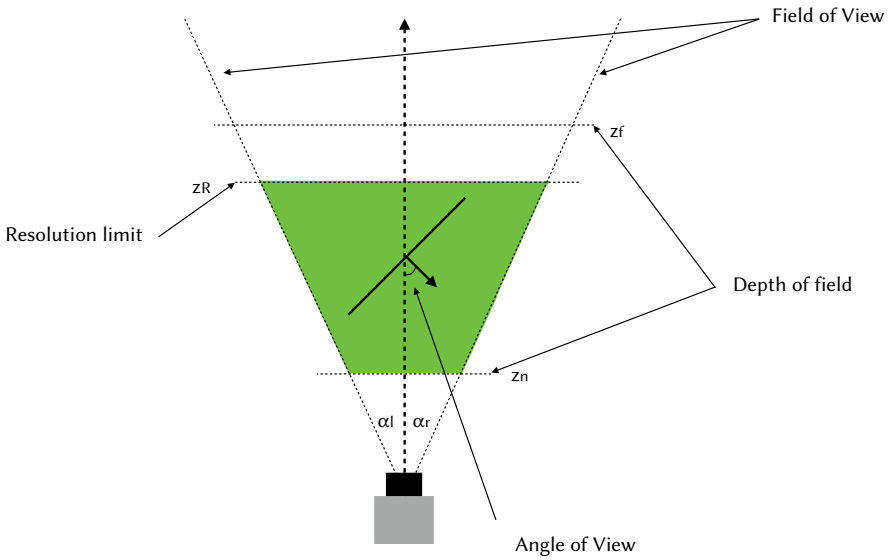



**Figure 2.5:** The components of the coverage function as geometric constraints on the field of view of the camera (top view).

constraints limit the field of view of the camera creating the so called viewing frustum. The frustum represents the volume of the scene, which is visible from the camera, that satisfies visibility, resolution,

and focus as well.

The camera's viewing frustum can be described with seven parameters: The four angles of the field of view, the near and far limits of the depth of field, and the far limit of the minimum resolution.

Given the focal length $f$ (mm), the width $w$ and height $h$ ($pixel$) of the sensor, the width $s_u$ and height $s_v$ (mm) of a pixel in the sensor, and the coordinates ($pixel$) of the optical centre $r_u$ and $r_v$; the four angles of the field of view, as measured from the optical axis, are given by

$$\alpha_l = \arctan \frac{r_u s_u}{f} \tag{2.2}$$

$$\alpha_r = \arctan \frac{(w - r_u) s_u}{f} \tag{2.3}$$

$$\alpha_t = \arctan \frac{r_v s_v}{f} \tag{2.4}$$

$$\alpha_b = \arctan \frac{(h - r_v) s_v}{f} \tag{2.5}$$

where $\alpha_l$, $\alpha_r$, $\alpha_t$, and $\alpha_b$ are the left, right, top, and bottom angles of the field of view, respectively.

The near $z_n$ and far $z_f$ limits of the depth of field are determined as follows

$$z_n = \frac{A f z_S}{A f + c \min(s_u, s_v)(z_S - f)} \tag{2.6}$$

$$z_f = \frac{A f z_S}{A f - c \min(s_u, s_v)(z_S - f)} \tag{2.7}$$

where $A$ is the diameter of the iris' aperture. $z_S$ is the distance along the optical axis at which the image is in focus. $c$ is the task's maximum blur ($pixel$).

Finally, the maximum distance that can be allowed before the resolution falls below the desired requirement is given by

$$z_R = R_{i/a} \min\left(\frac{w}{\tan \alpha_l + \tan \alpha_r}, \frac{h}{\tan \alpha_t + \tan \alpha_b}\right) \tag{2.8}$$

where $R_{i/a}$ is the task's minimum resolution in units of length per pixel.

The coverage model was developed by Mavrinac et al. [78]. This dissertation uses this model's metric of performance as the means to provide an objective way to validate the various simulations presented throughout this dissertation. The validation of this model is provided in Mavrinac [4], and Alarcon et al. [79].

# CHAPTER 3

# Graph-Based Sensor Deployment

> If you have built castles in the air, your work need not be lost; that is where they should be. Now put the foundations under them.
>
> Henry David Thoreau (1817–1862)

## 3.1 Overview

The design of vision systems is, in most cases, executed by a vision specialist, who will integrate a solution using off-the-shelf cameras and optics. The crucial part of the design is to develop a camera layout, that is, deciding on the position and orientation of the camera or cameras such that the inspection scene is imaged with sufficient quality. Even when using the device's manufacturer's guidelines for positioning, these are only approximations that leave out important details about the vision system for the sake of generality and compatibility. In consequence the vision specialists find themselves performing trial and error to refine the solution. This is at best a difficult task, and sometimes prohibitive because of the costs of implementing a camera layout for the sole purpose of testing it. Thus, it is desirable to have access to an automatic method for camera deployment.

The objective of this section is to provide vision specialists with a tool for automatically producing camera layouts for industrial inspection. For simplicity of presentation, at this point this tool is referred to as a black box with the following inputs and outputs in Table 3.1, which will be described in more detail in Section 3.5.

**Table 3.1**: Design Tool

| Inputs | Outputs |
|---|---|
| Task model | List of camera poses |
| Sensor-lens list | Expected performance metric |

## 3.2 The Problem

Given an ordered list of directional scene-points **p** and an ordered list of viewpoints **v** representing feasible camera poses, find the minimum number of viewpoints that maximizes the coverage of the scene-points. The formal problem is described as an integer programming problem.

$$\text{minimize} \sum_{j=0}^{|\mathbf{v}|-1} x_j, \quad \text{subject to} \tag{3.1}$$

$$\sum_{j=0}^{|\mathbf{v}|-1} \sum_{i=0}^{|\mathbf{p}|-1} \frac{C(v_j, p_i)x_j}{|\mathbf{p}|} \geq 1 \tag{3.2}$$

$$x_j \in \{0, 1\}; \quad j = 0, 1, \ldots, |\mathbf{v}| - 1 \tag{3.3}$$

equation (3.3) is a binary condition over the list **v**, meaning that the value of $x_j$ is 1 if viewpoint $v_j$ is in the solution, and 0 otherwise. (3.2) sets a constraint on the coverage strength for all points so that the coverage performance becomes 1. Note the subscript $i$ is reserved for scene-points, subscript $j$ is reserved for viewpoints, and $|\cdot|$ represents the length of the list.

## 3.3 The Solution Space

The size of the solution space is indeed very large, and it is in fact $2^{|\mathbf{v}|}$. An option could be, for instance, to reduce the size of the solution space by removing all binary combinations of the elements in **x** that are larger than the minimum necessary number of viewpoints. However, there is no *a priori* way of computing this number. In addition, a brute force algorithm that tests all possibilities would be untraceable because its order of growth is, at best, exponential with respect to the size of the viewpoint list.

### 3.3.1 Scene Discretization

A scene-point list is a prerequisite for the computation of the viewpoint list. The scene-point list can be obtained directly from the model of the task. In general, the task can be modelled by a triangular mesh, which in most cases comes in the form of a CAD model. The task might also be modelled as a point could; however, in such case a triangulation or tessellation operation must be applied first in order to estimate the surface normals of the task.

From the task model, the centroid of each triangle and the normal vector to the plane in which the triangle lies are computed. Using each of the triangles' centroids and normals, an ordered list of directional points is created. In this work it is assumed that the list of directional points effectively models the geometry of the task, and that the tringles are small enough to approximate the surface of the task with sufficient accuracy.

### 3.3.2 Viewpoint Generation

The solution space is populated only with feasible viewpoint candidates. Thus, an approach similar to that used by Tarbox and Gottschlich [80] is used. For every point in the scene-point list, an optimal viewpoint can be generated using the guidelines depicted in Figure 3.1.

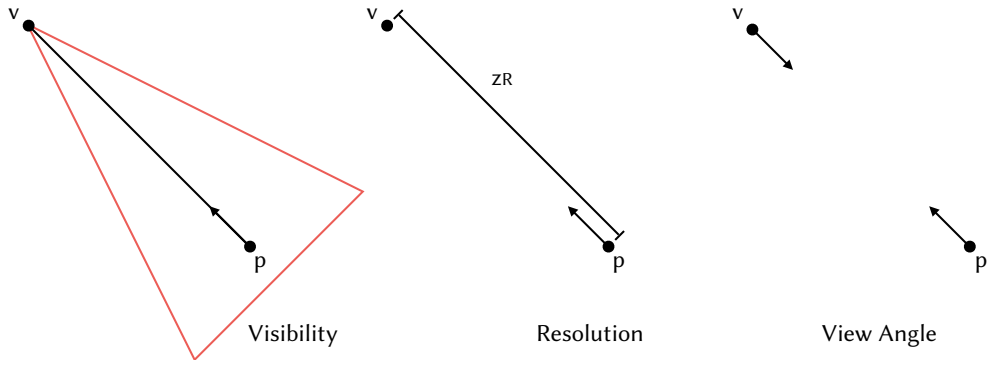



**Figure 3.1**: Viewpoint generation guidelines.

The explanations for these guidelines are as follows. First, a scene-point located along the optical axis maps to the centre of the image, which ensures maximum visibility. Second, resolution is a function of distance and it measures the units of length represented by a pixel in the image. The standoff distance $z_R$ is the furthest a viewpoint can be before resolution falls below a predetermined threshold. Finally, let $\mathcal{P}$ be the plane tangent to the surface where the scene-point lies, and let $\mathcal{I}$ be the image plane associated with the viewpoint. A scene-point with a normal that is aligned with the optical axis of the viewpoint is equivalent to having planes $\mathcal{P}$ and $\mathcal{I}$ be parallel with respect to each other and thus avoid skewed images and self occlusion. Equations (2.8) and (3.4a) to (3.5b) show how to compute the viewpoint space.

The position component.

$$v_x = p_x + z_R \sin(p_\rho) \cos(p_\eta) \tag{3.4a}$$
$$v_y = p_y + z_R \sin(p_\rho) \sin(p_\eta) \tag{3.4b}$$
$$v_z = p_z + z_R \cos(p_\rho) \tag{3.4c}$$

The direction component.

$$v_\rho = p_\rho + \pi \tag{3.5a}$$
$$v_\eta = p_\eta \tag{3.5b}$$

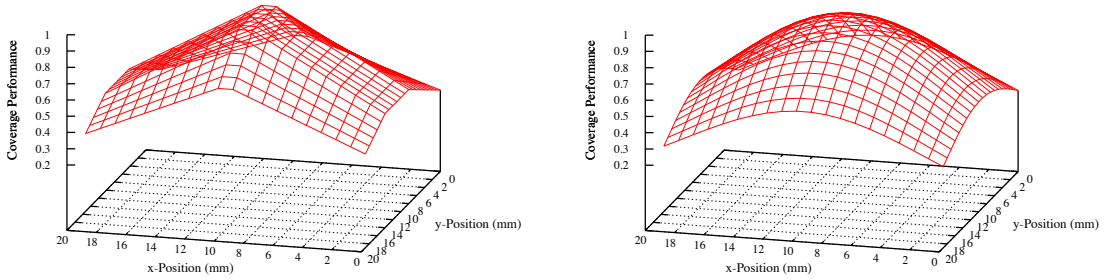
where scene-point $p$ is defined as $p = [x, y, z, \rho, \eta]$, and similarly for viewpoint $v$. In both cases $x$, $y$, and $z$ represent the position, whereas $\rho$ and $\eta$ represent the direction formatted in spherical coordinates.

### 3.3.3 Quality of the Solution Space

Next, evidence that the solution space has some optimal properties is provided. Although the response of the coverage function $C(v,p)$ is non-linear and very difficult to optimize (see Figure 5.2(a)), each viewpoint in the solution space is generated locally for each scene-point (see Section 3.3.2). It is in this context that the response of the coverage function becomes convex with respect to the plane in which the scene-point is located. When the scene is restricted to a plane, as it is the case with each triangle in the task model, it is possible to fit a smooth convex function to the response of the coverage function. An example of this can be seen in Figure 3.2 were the fit is given by

$$C_{fit} = e^{-\left(\frac{(x-x_o)^2}{2\sigma_x^2} + \frac{(y-y_o)^2}{2\sigma_y^2}\right)} \tag{3.6}$$

where $C_{fit}$ determines the coverage strength provided by a viewpoint at point $p = [x,y]$ in the viewpoint's local frame. The deviations, $\sigma_x$ and $\sigma_y$, determine the shape, whereas $x_o$ and $y_o$ determine the centroid of the polygon[1].




(a) Coverage Function (b) Gaussian Fit

**Figure 3.2**: Coverage Response – A comparison between the response of the coverage model and the response of the gaussian fit.

---

**1 Proposition (The Optimality of a Triangle's Centre)**
*The scene-point that yields an optimal viewpoint for visual coverage of a convex polygon is located at the centroid of the polygon.*

Proof The optimal viewpoint in a convex polygon is found by making the first derivative of the coverage function equal to zero. This is done, respectively, for $C'_{fitx}$ and $C'_{fity}$.

$$\frac{\partial}{\partial_x} C_{fit} = -\frac{(x-x_o)}{\sigma_x^2} e^{-\left(\frac{(x-x_o)^2}{2\sigma_x^2} + \frac{(y-y_o)^2}{2\sigma_y^2}\right)} \tag{3.7a}$$

$$\frac{\partial}{\partial_y} C_{fit} = -\frac{(y-y_o)}{\sigma_y^2} e^{-\left(\frac{(x-x_o)^2}{2\sigma_x^2} + \frac{(y-y_o)^2}{2\sigma_y^2}\right)} \tag{3.7b}$$

[1]By introducing $x_o$ and $y_o$ the Gaussian fit looses some generality and becomes applicable only for convex and fairly symmetrical polygons. In the case of the triangular mesh, the Gaussian fit is still valid.

which, by setting $\frac{\partial}{\partial_x} C_{fit} = 0$ and $\frac{\partial}{\partial_y} C_{fit} = 0$, yield

$$x = x_o \tag{3.8a}$$

$$y = y_o \tag{3.8b}$$

which indicates that the optimal viewpoint for a convex polygon is located at the centre of the polygon. ■

---

From the previous analysis –assuming that the triangles in the task model are at least smaller than the field of view of the camera– it can be concluded that a scene-point can effectively replace the triangle as a model of the task, and that the solution space has the property that every scene-point is optimally covered by at least one viewpoint.

The previous proof works well under the assumption that the Gaussian fit in (3.6) approximates the response of the coverage model within acceptable tolerance. As can be seen in Figure 3.2, the Gaussian fit does in fact approximate the coverage model for all convex-symmetric polygons. The method presented in this section is restricted to triangular meshes with fairly symmetrical triangles. If this is not the case, then an additional triangulation step can be applied to further partition the mesh.

### 3.3.4 Extended Solution Space

All the possible combinations of viewpoints from the viewpoint list –in which any viewpoint may or may not be present in the solution– represent the solution space. Because the solution space is composed of only one viewpoint for each triangle in the task model, one can argue that this reduction is too restrictive and not representative enough of its continuous counterpart[2]. In order to improve this, a partitioning of the triangular mesh into convex regions is performed by applying graph tools to traverse the topology of the mesh.

---

**1 Definition (Convex Region)**
*A convex region is a set of triangles where the vertices form a connected graph in which each triangle is defined by a cycle of length three, and the angle between the normals of any two adjacent triangles is in the range* $(0, \frac{\pi}{2}]$.

---

The convex regions are found by checking for each vertex in the mesh all the triangles that touch this vertex. If all the triangles satisfy the angle condition a convex region is found, and the vertices in the region are removed for the remainder of the search. Afterwards, each convex region is represented by a set of directional points. Since the normals associated to the points are unit vectors expressed in a common frame, a cone is fitted by averaging the normals and thus finding a viewpoint that is well posed to cover all triangles in the convex region. A new viewpoint is added to the solution space for each convex region.

[2]The optimal solution may lay in the space between the centre of two or more triangles.

The increased size of the solution space produces an increase in the computational cost for the viewpoint selection method. This is handled as follows. Firstly, the mesh is partitioned in linear time, using the so called *render dynamic*, in order to represent the topology of the mesh. The render dynamic data structure stores a set of lists representing all the relationships between the vertices, the edges, and the faces of the mesh allowing topology traversing in constant time. Lastly, occlusion computation can be avoided between any point in a convex region since convexity ensures that all triangles in the convex region are visible from the corresponding viewpoint's pose.

## 3.4 Optics Optimization

Equations (2.2) to (2.8) show how to compute the standoff distance for each viewpoint. However, these equations require that the focal length $f$ and the other internal parameters of the camera be fixed. In other words, before computing the viewpoint candidates it is necessary to select a camera sensor and lens.

### 3.4.1 Hardware Selection

The selection of camera sensors and lenses typically involves the consideration of five parameters: the sensor size, pixel pitch, image format, focal length, and lens aperture. For simplicity and generality only two parameters are considered[3], namely, the sensor size $[w, h]$ and the focal length $f$.

The performance of sensor networks is optimized by maximizing visual coverage and image quality. The performance is related to the angle of view, visibility, resolution, and focus. The position and orientation of the viewpoint determine the angle of view, and the visibility (to some extent). Furthermore, the properties of the sensor and lens determine visibility, resolution, and focus. Figure 3.3 shows the calibration of the focal length $f$ of a lens with variable zoom and focus settings. Figure 3.3 was computed by following the methods proposed in [81, 82, 83]. Although the behaviour of the focal length is highly non-linear, the general trend shows that as the zoom increases, so does the focal length.

---

2 **Proposition (Resolution vs Focal Length)**
*The resolution at which points in the scene map onto the image plane increases as the focal length increases.*

Proof Equation (2.8) can be rewritten as follows

$$R = d_S \max\left(\frac{\tan\alpha_l + \tan\alpha_r}{w}, \frac{\tan\alpha_t + \tan\alpha_b}{h}\right) \tag{3.9}$$

then, substituting (2.2) to (2.5), respectively, in (3.9)

$$R = d_S \max\left(\frac{\frac{r_u s_u}{f} + \frac{(w-r_u)s_u}{f}}{w}, \frac{\frac{r_v s_v}{f} + \frac{(h-r_v)s_v}{f}}{h}\right) \tag{3.10}$$

[3]The other three parameters can be calculated using the sensor size, the focal length, and the task parameters (see Mavrinac [4]).

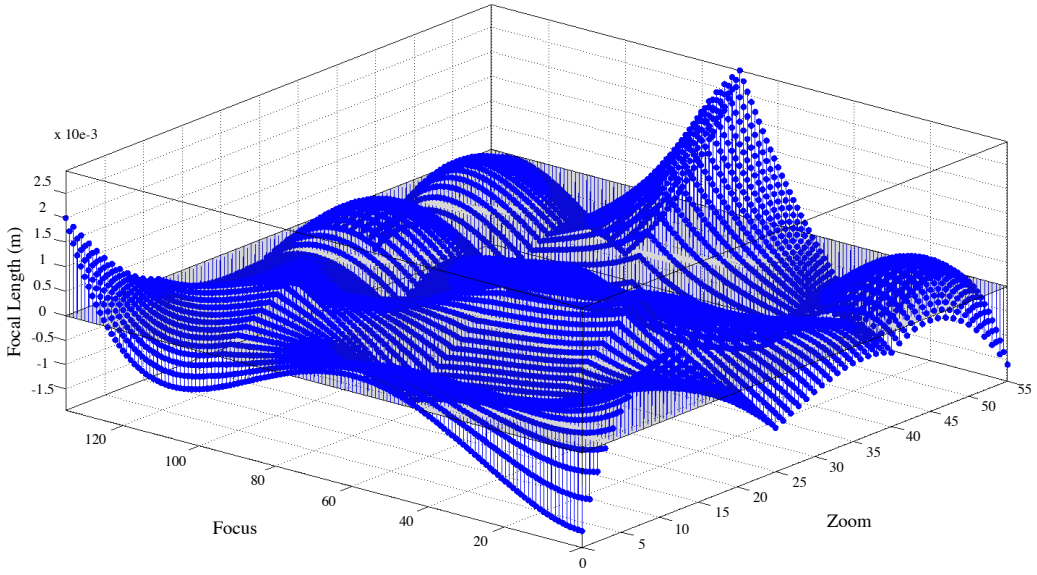



**Figure 3.3:** Calibration of variable lens.

further simplifying (3.10) with some algebraic manipulation, the resolution is given by

$$R = \frac{d_S}{f} \max(s_u, s_v) \tag{3.11}$$

Since resolution increases as the number of units of length per pixel decreases, (3.11) shows that the resolution is inversely proportional to the pixel size and directly proportional to the focal length.■

---

From (3.10) and (2.2) to (2.5), it can be seen that the resolution increases as the focal length increases and the sensor density increases; the later can be increased by increasing the sensor size and/or by decreasing the pixel pitch. Intuitively, one could argue that selecting the largest sensor and the lens with the largest focal length would provide the best possible resolution. However, this poses a problem for the visibility of the viewpoint; as (2.2) to (2.5) show: the size of the field of view is inversely proportional to the focal length. Thus increasing the focal length ultimately results in less coverage.

Since the only information available to compute the optimal size of the field of view is that which is being computed in the first place; this method resorts to having the user input a list of camera sensors and lenses. The tradeoff between resolution and visibility is resolved by simply applying the following steps:

1. *Field of view.* For all combinations of sensors and lenses from the list, the lower and upper limits of the angles of the field of view can be computed. Since the smallest angles yield the best resolution and the smallest coverage. And since the largest angles yield the worst resolution and the largest coverage, the angles of the field of view are simply set to be the mean between the two limits.

2. *Sensor size.* The size of the sensor is directly proportional to both the resolution and the size of the field of view, thus the largest sensor from the list of sensors is selected.

3. *Focal length.* With a fixed sensor and field of view, the focal length can be computed by assuming the optical centre is at the centre of the image and rearranging (2.2) to (2.5)

$$f = \frac{ws_u}{2\tan\frac{\alpha_h}{2}} = \frac{hs_v}{2\tan\frac{\alpha_v}{2}} \tag{3.12}$$

where $\alpha_h = \alpha_l + \alpha_r$ and $\alpha_v = \alpha_t + \alpha_b$ are the horizontal and vertical angles of the field of view.

Once the sensor has been selected and the focal length has been computed, the list can be discarded. The lens with the focal length that is closest to the computed value is selected.

## 3.5 The Sensor Deployment Algorithm

### 3.5.1 Graph-Based Data Structure

Given the ordered list of scene-points **p** and the associated list of viewpoints **v**, a data structure is constructed that will be helpful in formalizing the viewpoint selection algorithm later in this Section. The interaction topology between scene-points and viewpoints is represented using a graph $G = (V, E, w)$ with the set of vertices $V = \{p_i, \cdots, p_{|\mathbf{p}|}, v_j, \cdots, v_{|\mathbf{v}|}\}$ for $i = 0, \cdots, |\mathbf{p}| - 1$ and $j = 0, \cdots, |\mathbf{v}| - 1$, and edges $E \subseteq V \times V$. An edge exists between two vertices if one of the vertices is visible from the other. Additionally, the weight of the edge is equal to the coverage strength resulting from evaluating the pair of vertices in the edge, $w = C(v_j, p_i)$. It is evident by now that $G$ is a directed graph because only viewpoints can have visibility, in other words, $C(p_i, v_j) = 0 \ \forall\, i, j$ and $C(p_i, p_j) = 0 \ \forall\, i, j$. To illustrate this, Figure 3.4 shows the adjacency matrix of vision graph $G$.

| | $v_1$ | $\cdots$ | $v_{|\mathbf{v}|}$ |
|---|---|---|---|
| $p_1$ | $C(v_1, p_1)$ | $\cdots$ | $C(v_{|\mathbf{v}|}, p_1)$ |
| $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ |
| $p_{|\mathbf{p}|}$ | $C(v_1, p_{|\mathbf{p}|})$ | $\cdots$ | $C(v_{|\mathbf{v}|}, p_{|\mathbf{p}|})$ |

**Figure 3.4:** Adjacency matrix of vision graph $G$

Note that although viewpoints could, in theory, see other viewpoints, there is no use in a camera configuration where viewpoints are part of the scene. This is avoided by setting $C(v_i, v_j) = 0 \ \forall\, i, j$. The data structure is then the $|\mathbf{p}|$ by $|\mathbf{v}|$ resulting matrix. This matrix is very similar to the *measurability matrix* used by Scott [15]. One important difference between the work by Scott and this work, is that in [15] the viewpoint selection is solved by an algorithm that directly operates over the entries of the measurability matrix. In this work, this matrix is used as a first step to building a graph of visual overlap. For the remaining of this section this matrix will be referred to as the vision matrix $M$.

In order to handle and extract some useful information from the vision matrix, some tools are presented for computing the coverage of each viewpoint and the degree of overlap between viewpoints. The list of points that are covered from viewpoint $v_j$ is the column vector $M[:, j]$. The coverage of a single viewpoint $c(v_j)$ is defined as the coverage strength provided by that viewpoint for all scene points.

$$c(v_j) = \frac{1}{|\mathbf{p}|} \sum_{i=0}^{|\mathbf{p}|-1} M[i, j] \tag{3.13}$$

The maximum coverage $c_{\max}$ for all viewpoints in $M$ is

$$c_{\max} = \max\Big(c(v_j)\Big)\Big|_{\forall v_j \in \mathbf{v}} \tag{3.14}$$

For any two viewpoints $v_j$ and $v_k$ the degree of coverage overlap is defined as the coverage of the dot product of the two column vectors $M[:, j]$ and $M[:, k]$, normalized by the maximum coverage $c_{\max}$.

$$o(v_j, v_k) = \frac{\frac{1}{|\mathbf{p}|} \sum_{i=0}^{|\mathbf{p}|-1} \Big[M[i, j] \cdot M[i, k]\Big]}{c_{\max}} \tag{3.15}$$

### 3.5.2 Viewpoint Selection

The motivation for building this vision matrix, is that it enables the quantification of the degree of coverage $c(v_j)$ that each viewpoint candidate can provide. Additionally, using the degree of visual overlap $o(v_j, v_k)$, the system's graph of coverage overlap can be constructed. Let $O = (V, E, w)$ be the weighted undirected overlap graph, with the set of vertices $V = \{v_j, \cdots, v_{|\mathbf{v}|}\}$ for $j = 0, \cdots, |\mathbf{v}| - 1$, and edges $E \subseteq V \times V$. There exists an edge between a pair $v_j$ and $v_k$ if the viewpoints simultaneously cover some of the scene-points. The weight $w$ is equal to the overlap as given by (3.15).

Using both overlap graph $O$ and the row vector of coverage $\Big[c(v_j)\Big]\Big|_{\forall v_j \in \mathbf{v}}$, the compounded degree for each vertex in $O$ is computed. The compounded degree $d^c$ of a vertex $v_j \in O$ is defined as the product between the sum of the weights of all the edges that contain said vertex and the corresponding coverage $c(v_j)$, which is given by

$$d_j^c = c(v_j) \sum_{k=0}^{|h_j|-1} w_{jk} \tag{3.16}$$

where $k$ is used to iterate over the list, $h_j$, of neighbours of $v_j$, and $w_{jk}$ is the weight of the corresponding edge or overlap degree.

The premise of the viewpoint selection method is the following. Let $o(v_j, v_k) = 1.0$ be the overlap shared by viewpoints $v_j$ and $v_k$. Since the overlap is non-zero, any viewpoint of the two performs as good as the other at covering the shared scene. Furthermore, if $v_j$ has non-zero overlap with other viewpoints, then $v_j$ becomes a desirable viewpoint for selection since it can potentially replace its neighbours. It is important to be careful when using this criteria for viewpoint selection since a high degree of overlap does not guarantee a high degree of coverage. This is why, in (3.16), the sum of the weights is multiplied by the coverage.

**Algorithm 1** Graph-Based Viewpoint Placement

**Input:** $\mathbf{p}$
**Output:** $S$

From $\mathbf{p}$, generate $\mathbf{v}$ as given by (2.8) and (3.4a) to (3.5b)
**for** $i = 0 \rightarrow |\mathbf{p}|$ **do**
  **for** $j = 0 \rightarrow |\mathbf{v}|$ **do**
    $M[i, j] = C(v_j, p_i)$
  **end for**
**end for**
$V \leftarrow \mathbf{v}$
pairs ← non-repeated combinations of $V$
**for** pair ∈ pairs **do**
  append $E \leftarrow$ pair
  append $w \leftarrow o$(pair), as given by (3.15)
**end for**
$O = (V, E, w)$
**for** $v_j \in V$ **do**
  append $D^c \leftarrow d_j^c$, as given by (3.16)
**end for**
sort $D^c$ in descending order
**while** $|D^c| > 0$ **do**
  pivot ← $D^c[0]$
  append $S \leftarrow pivot$
  remove $D^c[0]$ from $D^c$
  N ← neighbours of pivot in $O$
  **for** neighbour ∈ neighbours **do**
    **if** neighbour ∈ $D^c$ **then**
      remove neighbour from $D^c$
    **end if**
  **end for**
**end while**
**return** $S$

A greedy algorithm is implemented in order to perform viewpoint selection. A list of the compounded degrees of all vertices in $O$ is ordered in descending order with respect to $d^c$. The algorithm starts by selecting the viewpoint with the highest degree, and proceeds to eliminate from the list all the other viewpoints that are direct neighbours in the overlap graph. This is repeated until the original list of compounded degrees is empty. The motivation for eliminating neighbouring viewpoints is that this process reduces redundancies (i.e. excessive overlap), and thus maximizes the area being covered by the viewpoints. Algorithm 1 shows the viewpoint placement process.

### 3.5.3 Cost Minimization

After performing the viewpoint placement, it is necessary to further minimize the number of viewpoints. A binary search algorithm that operates over the list of selected viewpoints is used, as

**Algorithm 2** Viewpoint Minimization

**Input:** $S$
**Output:** $temp, f$

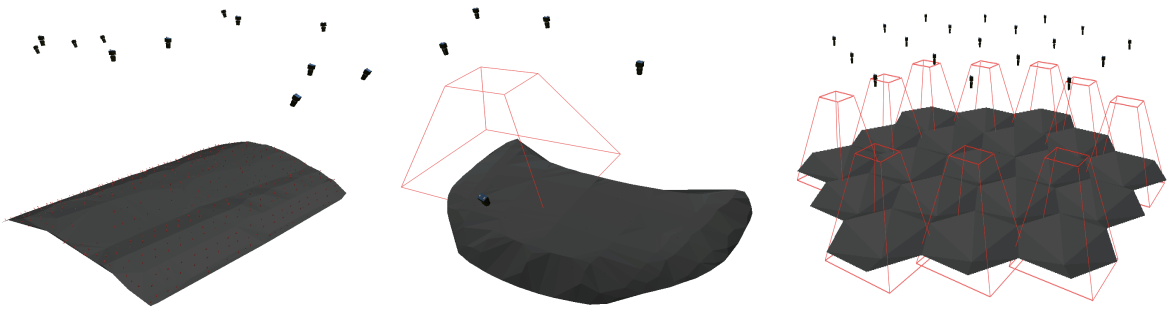
```
best ← F(S),  as given by (2.1)
bestNumber ← |S|
cut ← 1/2
temp ← S[0 : (|S| · cut)]
while |temp| ≠ bestNumber do
    f ← F(temp)
    if abs(best − f) < ε then
        S ← S[0 : (|S| · cut)]
        cut ← 1/2
        bestNumber ← |S|
    else
        cut ← 3/4
    end if
    temp ← S[0 : (|S| · cut)]
end while
return temp, f
```

given in Algorithm 1. The binary search works as follows. Given a list of viewpoints, the list is cut in half and its coverage performance $F(C, R)$ tested, if the performance remains within a predefined threshold $\varepsilon$ then the algorithm continues to cut the list by half. On the other hand, if the performance falls below $\varepsilon$ then the cut is increased (i.e. selecting more than half of the viewpoints). The algorithm stops when the minimum number of viewpoints that can perform relatively close to the original is found. It is important to note that a binary search can be used only if the list of viewpoints is ordered with respect to the coverage performance. That is, there is a direct relationship between the number of cameras and the coverage performance of the list. This is true in this case because the viewpoint placement algorithm sorted the list of viewpoints based on the compounded degree, which is directly related to the coverage. The binary search is implemented in Algorithm 2.

## 3.6 Experimental Validation

### 3.6.1 Simulations

In this simulation the graph-based method is used to deploy a sensor network for three different polygonal meshes. For each experiment, the results obtained using the solution space and the extended solution space, as described in Section 3.3, are compared. In this case the sensor networks are deployed and tested by simulation in Adolphus; the performance of the deployment is measured using the coverage performance (2.1). The results are summarized in Tables 3.2 to 3.4. The columns are described as follows. The second column shows the number of triangles in the polygonal mesh. The third column shows the number of cameras selected applying Algorithm 1. The fourth column shows the number of cameras selected after the viewpoint minimization as per Algorithm 2. The fifth column is the coverage performance as defined by Mavrinac [4]. Finally, the last column shows the



(a) Shape A (b) Shape B (c) Shape C

**Figure 3.5**: Graph-based camera network deployment.

time expressed in minutes.

**Table 3.2**: Graph-Based Simulation Results (shape A)

| Method | Triangles | Greedy | Optimal | Performance | Time |
|---|---|---|---|---|---|
| Normal | 379 | 10 | 8 | 0.9603 | 6.55 |
| Extended | 379 | 11 | 8 | 0.9603 | 6.89 |

**Table 3.3**: Graph-Based Simulation Results (shape B)

| Method | Triangles | Greedy | Optimal | Performance | Time |
|---|---|---|---|---|---|
| Normal | 418 | 19 | 10 | 0.9616 | 9.14 |
| Extended | 418 | 19 | 10 | 0.9616 | 8.71 |

**Table 3.4**: Graph-Based Simulation Results (shape C)

| Method | Triangles | Greedy | Optimal | Performance | Time |
|---|---|---|---|---|---|
| Normal | 114 | 24 | 24 | 0.7281 | 0.22 |
| Extended | 114 | 19 | 19 | 0.9474 | 0.23 |

From the results it can be seen that both methods compute a solution approximately under the same amount of time. When comparing the deployments for shapes A and B, it can be seen that the cost and performance are the same for both methods. However, in the case of shape C the method using the extended solution space outperforms the method using the regular solution space. This evidence is in accordance with the theoretical expectation (see Section Section 3.3). The method using the extended solution space performs equally well, or better than the method using the regular solution space.

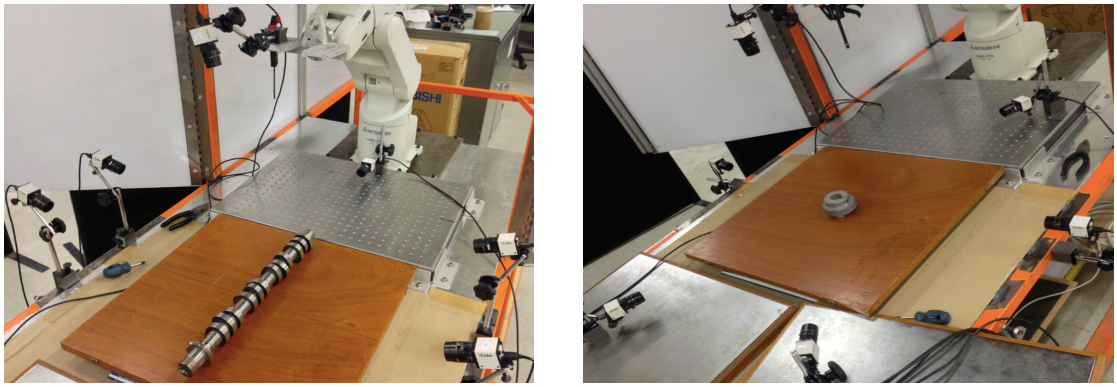
In previous work presented in [84, 85], Algorithm 1 had a shorter execution time. Occlusion checks for static objects in the scene have been implemented since, causing Algorithm 1 a longer execution time. On the other hand, the design tool for sensor deployment now implements the optics selection method described in Section 3.4. In previous work, the method for optics selection relied on a *generate-and-test* approach that added a linear time complexity to the overall execution time with

respect to the number of sensor-lens combinations in the input list. Thanks to the method described in Section 3.4 the selection of the optics is now done in constant time. Table 3.5 shows a comparison between the execution times using generate-and-test and the new analytic approach. In this case there are a total of five sensor-lens combinations in the input list.

**Table 3.5**: Execution Time Comparison for Graph-Based Method

| | Generate-and-test | Analytic approach |
|---|---|---|
| Shape A | 32.38 min | 6.89 min |
| Shape B | 42.67 min | 8.71 min |
| Shape C | 01.17 min | 0.23 min |

### 3.6.2 Experiments



(a) Crankshaft (b) Motor Coupling

**Figure 3.6**: Graph-based sensor network deployment.

Next, the graph-based method is tested using a physical apparatus. In this experiment, camera configurations are generated to cover two mechanical parts: a crankshaft and a motor coupling, as seen in Figure 3.6. In this case only one type of camera (iCube NS4133BU) and one type of lens (NET SV-0813V) are used; the task parameters were selected as follows: $R_i = R_a = 0.4$, $c_i = 1.0$, $c_a = 5.0$, and $\zeta_i = \zeta_a = 0.9599$.

There is a problem inherent in the positioning of the cameras. Unlike the simulation, human operators are unable to place the cameras at the exact location described by the generated sensor configuration. The solution is to use extrinsic camera calibration as means of feedback and to repeat the process of positioning and calibrating until all cameras are close enough to their respective locations. Although the robotic manipulator in Figure 3.6(b) would have been an ideal solution to this problem, the reach of the robotic arm is too small to effect all camera positions. In this experiment it is reported that the mean euclidean error between the camera locations and the actual locations is 10mm.

Using image processing operations the performance criteria is measured from the images. Criteria such as the total area of the part that is visible to the cameras, the mean resolution at which the

cameras image the part, and the greatest angle at which cameras can see non-parallel faces of the part. These measurements are reported in Tables 3.6 and 3.7 under "Measured". The data reported under "Expected" is the result of the simulation after the sensor configuration was generated.

**Table 3.6**: Crankshaft

| | **Expected** | **Measured** |
|---|---|---|
| **Visibility** | 96.58% | 91.94% |
| **Resolution** | 0.4mm/$px$ | 0.35mm/$px$ |
| **Angle** | 53.10° | 51.98° |

**Table 3.7**: Coupling

| | **Expected** | **Measured** |
|---|---|---|
| **Visibility** | 100% | 98.21% |
| **Resolution** | 0.4mm/$px$ | 0.31mm/$px$ |
| **Angle** | 54.74° | 54.00° |

From Tables 3.6 and 3.7 it can be seen that both the visibility and the angle of view of the final configuration were lower than expected, although still close. On the other hand, the measured resolution outperformed the expected one. Thus, the graph-based method for automatic viewpoint selection is able to perform while satisfying the visual requirement of resolution, which is of utmost important for industrial inspections.

# Part II

# Tensor Framework

CHAPTER 4

# Tensor Framework

> Whether you can observe a thing or not depends on the theory which you use. It is the theory which decides what can be observed.
>
> Albert Einstein (1879–1955)

The ultimate goal of modeling the vision system is to be able to predict the system's performance. This predictive quality may be useful for simulations and optimization, and in terms of the later, the optimal state is defined as follows. A triangle –the atomic unit that represents a task– is optimally imaged by the camera when its centre is at the centre of the viewing frustum, and the normal to its surface is collinear with respect to the optical axis. Figure 4.1 depicts this concept.

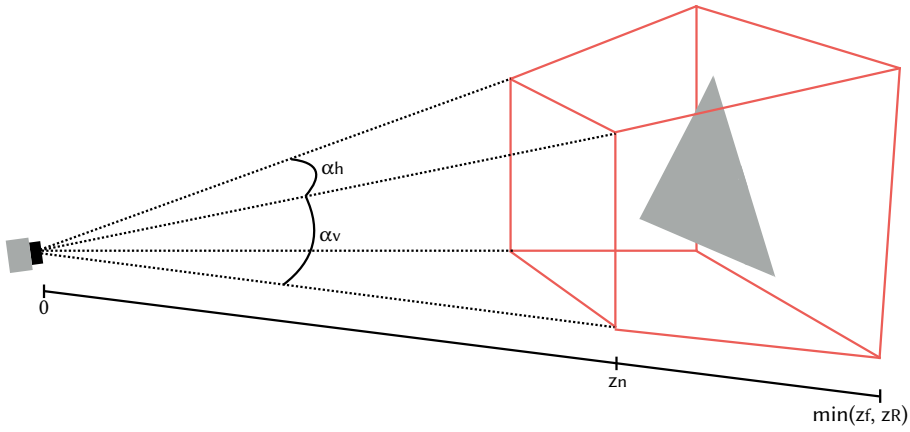



**Figure 4.1**: The camera's viewing frustum and a task represented by a single triangle.

The squared frustum in Figure 4.1 is a three dimensional representation of the one shown in Figure 2.5. Before introducing the vision distance two concepts need to be formalized: the camera's viewing frustum, and the triangle as the atomic unit that represents a task.

## 4.1 Camera Tensor

Using equations (2.2) to (2.8) the viewing frustum can be constructed by cutting a pyramid –with apex at the optical centre and angles equal to those of the field of view– at a distance of $z_n$ from the apex. The base is located at a distance of $\min(z_f, z_R)$ from the apex. In Figure 4.1 $\mathbf{p}_C$ represents the position of the frustum in the world frame.

The camera tensor is a $3 \times 3$ matrix, whose column vectors are the three vectors shown inside the viewing frustum in Figure 4.2 (expressed in the world coordinate frame). These three vectors form an orthogonal basis which fully describes the orientation of the frustum and whose lengths describe the shape of the frustum.

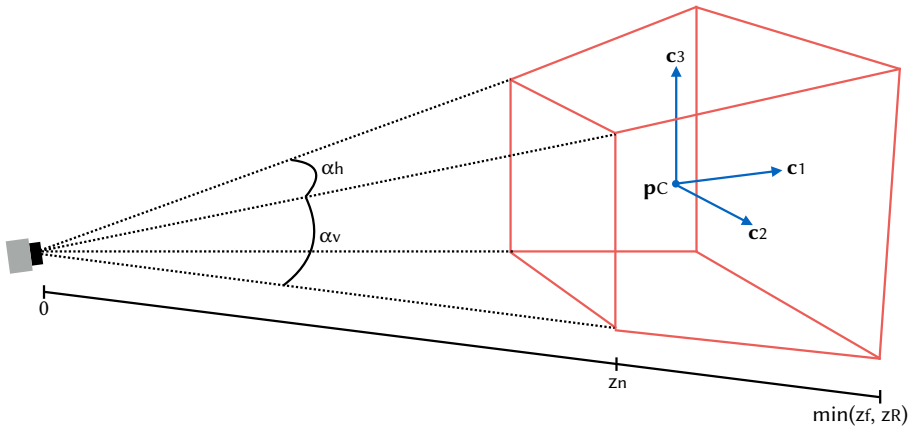



**Figure 4.2**: The camera tensor.

The motivation for representing the camera's frustum with a tensor –as opposed to a set of eight points in space– is that the tensor can also be expressed in a quadratic form, and thus allow for a larger set of optimization tools that can be applied. In geometrical terms, the camera's frustum is being modeled with an ellipsoid. The relationship between the camera tensor and the ellipsoid can be found by applying the principal axis theorem to the ellipsoid's equation [86].

$$c_{11}x^2 + (c_{12} + c_{21})xy + (c_{13} + c_{31})xz + c_{22}y^2 + (c_{23} + c_{32})yz + c_{33}z^2 = 1 \tag{4.1}$$

$$\mathbf{x}^T C \mathbf{x} = \begin{bmatrix} \mathbf{c}_1 & \mathbf{c}_2 & \mathbf{c}_3 \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \tag{4.2}$$

where $C$ is the camera tensor and $\mathbf{x}$ is the vector $[x\ y\ z]^T$ of components of the Euclidean space.

## 4.2 Triangle Tensor

In this dissertation the triangle is the atomic unit that represents a task, and the reasoning is twofold. First, a triangle is an entity that can carry information about the location and the orientation of a single feature point in the scene (i.e. the normal represents orientation and its centre represents location).

Second, in most industrial applications a task or target is represented as a triangular mesh (e.g. CAD model).

At the core of this formulation, the goal is to measure the visual distance between a camera and a triangle. In other words, it is necessary to quantify how close the camera is from imaging the triangle with optimal quality. In order to devise such distance metric, it is necessary to express the triangle in a form that is compatible with the camera tensor; thus the triangle tensor is introduced.

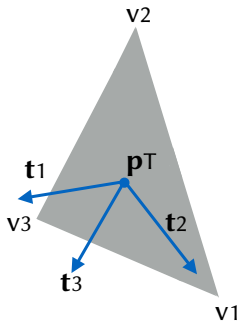



**Figure 4.3**: The triangle tensor.

Similarly to the camera tensor, an orthogonal basis can be used to describe the orientation of the triangle. Consider the vertices of the triangle in Figure 4.3 $v_1$, $v_2$, and $v_3$ as well as the centre of the triangle $\mathbf{p}_T$ (expressed in the world coordinate frame). The first column vector of the triangle tensor $\mathbf{t}_1$ is given by the cross product $\overrightarrow{\mathbf{p}_T v_1} \times \overrightarrow{\mathbf{p}_T v_2}$. The second column vector of the tensor $\mathbf{t}_2$ is $\overrightarrow{\mathbf{p}_T v_1}$. The third column vector of the tensor $\mathbf{t}_3$ is given by the cross product $\mathbf{t}_1 \times \mathbf{t}_2$. The tensor is given by

$$T = \begin{bmatrix} \mathbf{t}_1 & \mathbf{t}_2 & \mathbf{t}_3 \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix} \tag{4.3}$$

where $T$ is the triangle tensor.

At this point it is important to make a note on the camera and triangle tensors and how they approximate their respective visual counterparts. The camera tensor's orthogonal basis can be used to describe the orientation of the frustum, and the magnitudes of its column vectors can be used to approximate the volume of the frustum. In the case of the triangle tensor, column vectors $\mathbf{t}_2$ and $\mathbf{t}_3$ can be used to approximate the area of the triangle[1]. Finally, since tensors are independent of the coordinate frame used to describe them, one must keep track of the position, in the world coordinate frame, of the frustum's centre or the triangle's centre. In either case the position is denoted using the variable name $\mathbf{p}$ and the tensor's name as the subscript.

## 4.3 Tensor Operators

One of the goals of this dissertation is to provide a measure of closeness between two visual entities, in the Euclidean sense as well as the degree of alignment between their orientations. Next, some

[1]Column vector $\mathbf{t}_1$ is always expressed as a unit vector since it represents no physical property other than the direction of the normal to the triangle's surface.

tensor operations are defined before the Frobenius norm can be introduced as a way to compute the distance between two orientations.

Although the following operators can be applied to tensors of any dimension. This dissertation deals only with tensors in $\mathbb{R}^{3\times3}$. In all the following operations all tensors are assumed to be of size $3 \times 3$.

Given tensors $A$ and $B$, tensor $A$ is said to be equal to tensor $B$ according to the following

$$A = B \iff a_{ij} = b_{ij}, \quad \text{for } i,j \in [1,3] \tag{4.4}$$

Given tensor $A$, the negation operator is defined for a tensor as follows

$$-A = \begin{bmatrix} -\mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \end{bmatrix} \tag{4.5}$$

The reason from this particular definition is driven by the physical interpretation of the vision system and will be better explained later in Section 5.1.

By abuse of notation, a unit tensor is defined as a tensor whose column vectors are all unit vectors. Given tensor $A$, the unit tensor is given as

$$\frac{A}{||A||} = \begin{bmatrix} \frac{\mathbf{a}_1}{||\mathbf{a}_1||} & \frac{\mathbf{a}_2}{||\mathbf{a}_2||} & \frac{\mathbf{a}_3}{||\mathbf{a}_3||} \end{bmatrix} \tag{4.6}$$

where $||\mathbf{a}_n|| = \sqrt{\mathbf{a}_n^T \mathbf{a}_n}$, for $n = 1, 2, 3$

The norm of a tensor is defined as the geometric mean of the eigenvalues of the tensor. This is termed the geometric norm of a tensor, and it is given by

$$||A||_G = \sqrt[3]{\lambda_1 \cdot \lambda_2 \cdot \lambda_3} \tag{4.7}$$

where $\lambda_1$, $\lambda_2$, and $\lambda_3$ are the eigenvalues of $A$.

Given $A$ and $B$, the tensor product $A \otimes B$ is defined as the Kronecker product, and it is given by

$$A \otimes B = \begin{bmatrix} a_{11}\ B & a_{12}\ B & a_{13}\ B \\ a_{21}\ B & a_{22}\ B & a_{23}\ B \\ a_{31}\ B & a_{32}\ B & a_{33}\ B \end{bmatrix} \tag{4.8}$$

# CHAPTER 5

# Vision Distance

> Equations are just the boring part of mathematics. I attempt to see things in terms of geometry.
>
> Stephen Hawking (1942–)

In Chapter 4 the camera's frustum and triangle were described using a tensor and a point, namely $C$ and $\mathbf{p}_C$ for the camera and $T$ and $\mathbf{p}_T$ for the triangle. For simplicity, the tensor and the point are combined into a single representation of the camera and triangle as follows

The camera is represented by

$$\mathbf{C} = \begin{bmatrix} C & \mathbf{p}_C \\ \mathbf{0} & 1 \end{bmatrix} \tag{5.1}$$

and the triangle is represented by

$$\mathbf{T} = \begin{bmatrix} T & \mathbf{p}_T \\ \mathbf{0} & 1 \end{bmatrix} \tag{5.2}$$

where $C$ and $T$ are the camera and triangle tensors, respectively. $\mathbf{p}_C$ and $\mathbf{p}_T$ are the camera's frustum and triangle centres, respectively. $\mathbf{0} = [0, 0, 0]$.

It is important to note that under the framework presented in this dissertation, any normalized tensor satisfies the following conditions: $T^{-1} = T^T$, $\det(T) = 1$. Thus a normalized tensor is also a rotation matrix and a member of the special orthogonal group $SO(3)$.

The camera $\mathbf{C}$ and triangle $\mathbf{T}$ are $4 \times 4$ matrices. These matrices are defined for ease of presentation, but they could also be used to represent a pose in the special Euclidean group $SE(3)$, that is, position and orientation in homogeneous coordinates.

## 5.1 Frobenius Distance

An entry-wise norm treats an $m \times n$ matrix as an $mn$ vector, and then applies a vector norm. In the case of the $p$-norm the norm of a tensor is defined as

$$||A||_p = \left( \sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^p \right)^{1/p} \tag{5.3}$$

The Frobenius norm arises when $p = 2$ (see [87]), and in the case of the $3 \times 3$ tensor the Frobenius norm is given by

$$||A||_F = \sqrt{\sum_{i=1}^{3} \sum_{j=1}^{3} a_{ij}^2} \tag{5.4}$$

Based on (5.4), the Frobenius distance between two tensors is defined as

$$d_F(C,T) = \sqrt{\sum_{i=1}^{3} \sum_{j=1}^{3} (c_{ij} - t_{ij})^2} \tag{5.5}$$

---

3 **Proposition ($d_F(\cdot,\cdot)$ is a Metric on $SO(3)$)**
*The Frobenius distance $d_F(C,T)$ is a metric on $SO(3)$, $\forall\ C,T \in SO(3)$.*

Proof Since the Frobenius distance in (5.5) is induced from the Frobenius norm (5.4), which is defined for all elements in $\mathbb{R}^{3\times3}$ (see [87]). And since any element in $SO(3)$ is also an element in $\mathbb{R}^{3\times3}$. It follows that the Frobenius distance is a distance for all $C,\ T \in SO(3)$. ■

---

Since the camera and triangle tensors are, by construction, orthogonal bases, they represent each a rotation. However, in order for (5.5) to give a meaningful quantity of the difference between any two orthogonal tensors (or rotations), the tensors must be first normalized into unit tensors via (4.6). Additionally, from the physical interpretation of the vision system it is known that for a camera to image a triangle with optimal quality, the optical axis and the normal to the triangle's surface must be collinear but with opposite direction[1]. Since the distance metric should be 0 at the optimal point, it is necessary to negate either the optical axis or the normal to the triangle's surface. Recall that $\mathbf{c}_1$ and $\mathbf{t}_1$ represent the optical axis and the normal to the triangle's surface, respectively in $C$ and $T$.

Finally, since both the camera and triangle tensors are subject only to rigid three-dimensional Euclidean transformations, then only two of the three column vectors in an orthogonal tensor can have opposite directions. This last constraint gives an upper bound on the Frobenius distance between any two orthogonal unit tensors.

$$d_F\left( \frac{C}{||C||}, \frac{-T}{||T||} \right) \quad \in [0, \sqrt{8}] \tag{5.6}$$

[1]This is to avoid self-occlusion from the triangle itself.

**4 Proposition ($d_F(\cdot,\cdot)$ is Bounded between 0 and $\sqrt{8}$)**
*The Frobenius distance of any two rotation matrices in $SO(3)$ has a lower bound at 0 and an upper bound at $\sqrt{8}$.*

Proof Let $A$ and $B$ be arbitrary normalized tensors in $SO(3)$. The Frobenius distance is given as follows:

$$\begin{aligned} d_F(A,B) =& \|A-B\|_F \\ =& \sqrt{trace\left((A-B)^T(A-B)\right)} \\ =& \sqrt{trace\left(A^TA - A^TB - B^TA + B^TB\right)} \\ =& \sqrt{trace\left(2I\right) - trace\left(2A^TB\right)} \\ =& \sqrt{6 - trace\left(2A^TB\right)} \end{aligned} \tag{5.7}$$

Based on the property of rotation matrices, it is known that $C = A^TB$ is still a rotation matrix. Using the axis-angle representation, the rotation matrix $C$ can be rewritten as

$$C = \begin{bmatrix} \cos\theta + u_x^2(1-\cos\theta) & u_xu_y(1-\cos\theta) - u_z\sin\theta & u_xu_z(1-\cos\theta) + u_y\sin\theta \\ u_xu_y(1-\cos\theta) + u_z\sin\theta & \cos\theta + u_y^2(1-\cos\theta) & u_zu_y(1-\cos\theta) - u_x\sin\theta \\ u_xu_z(1-\cos\theta) - u_y\sin\theta & u_zu_y(1-\cos\theta) + u_x\sin\theta & \cos\theta + u_z^2(1-\cos\theta) \end{bmatrix} \tag{5.8}$$

where $\theta$ is the angle, and $[u_x\ u_y\ u_z]^T$ is the axis satisfying $u_x^2 + u_y^2 + u_z^2 = 1$. It is seen from (5.8) that

$$trace\left(C\right) = trace\left(A^TB\right) = 1 + 2\cos\theta \tag{5.9}$$

By substituting (5.9) into (5.7), it is shown that

$$d_F(A,B) = \sqrt{4 - 4\cos\theta} \tag{5.10}$$

Therefore, it is clear that the lower bound of the Frobenius distance $d_F(A,B)$ between any two rotation matrices is 0 and the upper bound is $\sqrt{8}$. Moreover, it is also interesting to find that the Frobenius distance between two rotation matrices merely depends the relative rotation angle, but independent of the rotation axis. ■

## 5.2 Vision Distance

Having already defined the necessary tools, the vision distance can now be introduced. The vision distance quantifies the closeness with which a given camera is to optimally image a given triangle. This is achieved by merging the Euclidean distance and the Frobenius distance; however, one of the two must be normalized first. Since the upper bound of the Euclidean distance lies at infinity, and the

Frobenius distance has both a lower and upper bounds, the Frobenius distance is normalized instead. The normalized Frobenius distance is given by

$$\frac{d_F\left(\frac{C}{||C||}, \frac{-T}{||T||}\right)}{\sqrt{8}} \quad \in [0, 1] \tag{5.11}$$

Finally, with optimization in mind, it is desirable that the vision distance exhibit the property of reducing to the Euclidean distance whenever the two tensor operands are closest to each other, as well as placing a heavy weight whenever the two tensor operands are furthest from one another. Thus (5.12) is termed the Frobenius-based orientation scaling factor

$$o_F = \frac{1}{1 - \frac{d_F\left(\frac{C}{||C||}, \frac{-T}{||T||}\right)}{\sqrt{8}}} \quad \in [1, \infty) \tag{5.12}$$

Since the Frobenius distance is a distance in $SO(3)$, and it is bounded to the range $[0, \sqrt{8}]$. It follows that the orientation scaling factor $o_F$ exhibits the property of being a scalar in the range $[1, \infty)$ for all $C,\ T \in SO(3)$.

The vision distance could be defined as

$$d_V(\mathbf{C}, \mathbf{T}) = o_F\sqrt{(\mathbf{p}_C - \mathbf{p}_T)^T(\mathbf{p}_C - \mathbf{p}_T)} \tag{5.13}$$

The vision distance in (5.13) works well except when the Euclidean distance $d_E(\mathbf{p}_C, \mathbf{p}_T) = 0$. In this case the information provided by the orientation scaling factor $o_F$ becomes irrelevant. One example where this problem arrises is when the triangle is at the centre of the viewing frustum but looking away from the camera. This case gives a vision distance of 0 but the triangle is self-occluded, and thus invisible to the camera.

To remediate the situation described above, an additive term is introduced to the Euclidean component of the vision distance, which is small enough to be negligible but still nonzero, and thus preventing $o_F$ from becoming irrelevant. The final version of the vision distance is defined as

$$d_V(\mathbf{C}, \mathbf{T}) = o_F\left(\sqrt{(\mathbf{p}_C - \mathbf{p}_T)^T(\mathbf{p}_C - \mathbf{p}_T)} \ + \ \varepsilon\right) \tag{5.14}$$

where $\varepsilon \to 0$.

---

5 **Proposition ($d_V(\cdot,\cdot)$ is a Metric on $\mathbb{R}^3$.)**
*The vision distance in (5.13) is a proper distance metric on $\mathbb{R}^3$.*

Proof The following proof is trivial and it is presented here for completeness. It is well known that the Euclidean distance $d_E(\mathbf{x}, \mathbf{y})$ is a metric on $\mathbb{R}^3$ for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$. Furthermore, $\alpha d_E(\mathbf{x}, \mathbf{y})$, $\alpha > 0$ is also a metric on $\mathbb{R}^3$ (see Naylor et al. [88]). By induction, it follows that the vision distance $o_F d_E(\mathbf{p}_C, \mathbf{p}_T)$, $o_F > 0$ is also a metric on $\mathbb{R}^3$. ■

---

Although the vision distance is a proper distance in $\mathbb{R}^3$, its operands are $\mathbf{C}$ and $\mathbf{T}$ in $SE(3)$. However, the vision distance is not a proper distance in $SE(3)$. The reason is that $o_F$, as opposed to $d_F(\cdot,\cdot)$, is not a distance in $SO(3)$ because it never becomes 0. Despite these facts, the vision distance still exhibits a behaviour similar to what one would expect from a distance in $SE(3)$. Evidence of this claim is shown in Section 5.3.1.

## 5.3 Validation of the Vision Distance

In the following experiments the vision distance is validated and its performance compared using both simulations and experiments. The comparison is possible thanks to a publicly available implementation of the work in Mavrinac [4]. The experimental work and simulations in this dissertation are done using the Adolphus simulation software [89]. Adolphus is free software licensed under the GNU General Public License (see Figure 5.1).

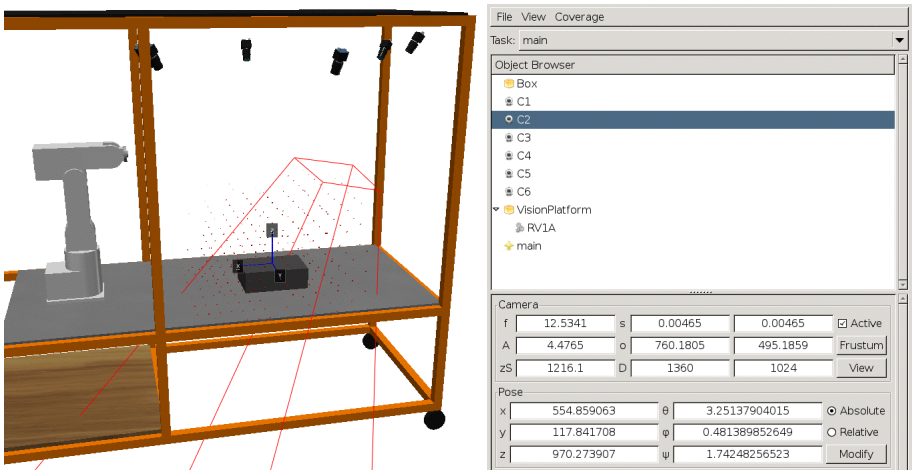



**Figure 5.1**: Simulation Software.

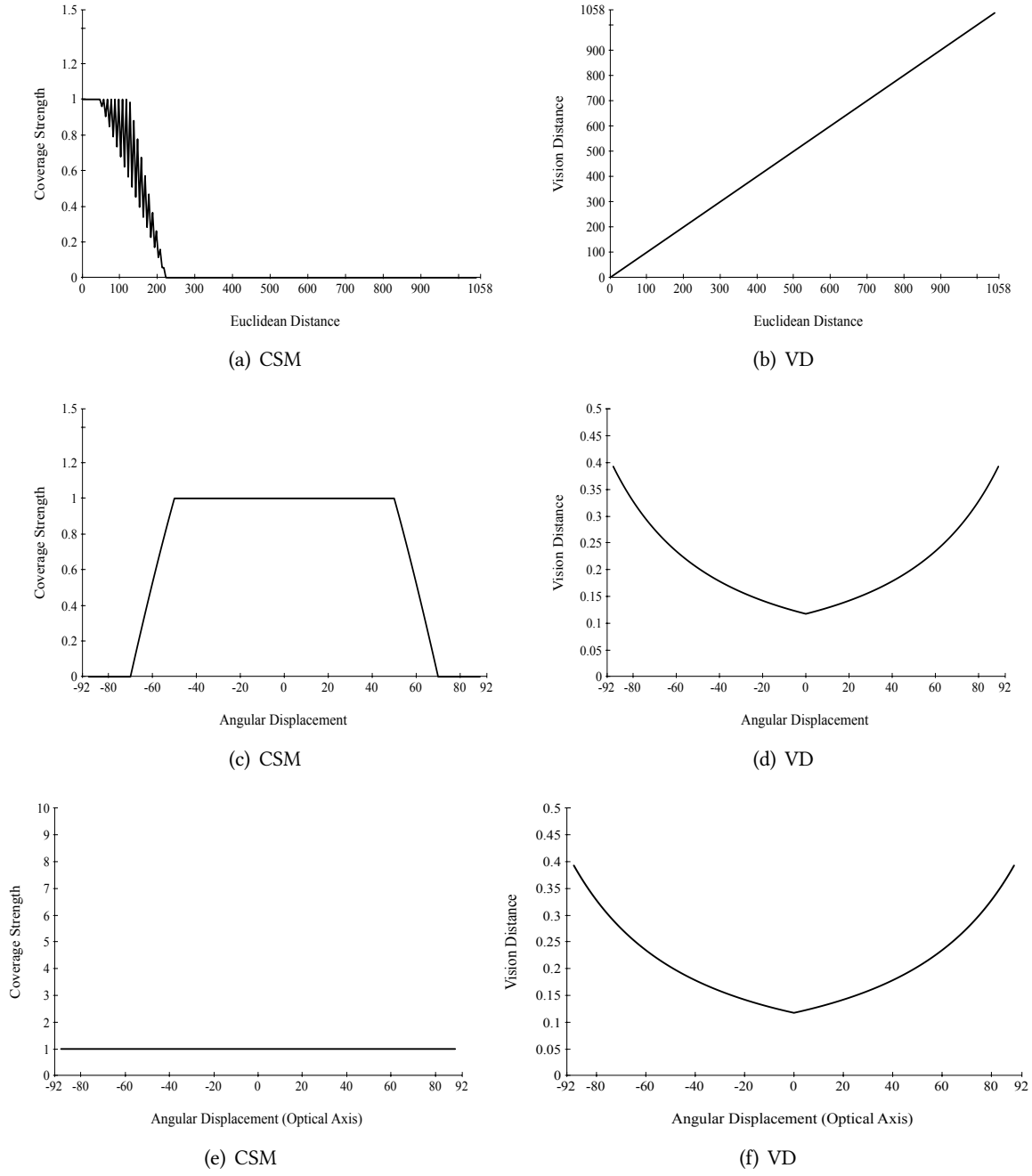
### 5.3.1 Simulations

In this simulation a simple scene consisting of a camera and a task is defined. The simulation is executed twice; once where there is only translational displacement of the task along the optical axis, and once where there is only rotational displacement of the task along the vertical axis. For each execution of the simulation, the response of both the coverage strength model and the vision distance is recoded. The results are shown in Figure 5.2.

Figure 5.2(a) shows the coverage performance against the absolute Euclidean distance. It is important to note that the coverage strength model is highly nonlinear as the existence of multiple local optimum is evident. Figure 5.2(b) shows that, when $o_F = 1$ and the movement is purely translational, the vision distance reduces to the Euclidean distance. The reduction to the Euclidean distance by the vision distance is evident in the linearity of Figure 5.2(b).

In the case of pure rotational movement the task rotates from a point of self-occlusion at $-90°$, then it passes through the optimal point, and rotates to self-occlusion at $90°$ (as measured from the optical axis). Figure 5.2(c) shows the coverage performance against the angular displacement, and Figure 5.2(d) shows the similar response for the vision distance. The trapezoidal shape of the coverage model's response shows the presence of multiple optimal points, where the vision distance shows a property much more helpful in optimization; the optimal point is unique.

In Figures 5.2(e) and 5.2(f) the task rotates around the optical axis of the camera. In this case there is no self-occlusion of the task; however, the task rotates around the optical axis in the image plane.




(a) CSM (b) VD

(c) CSM (d) VD

(e) CSM (f) VD

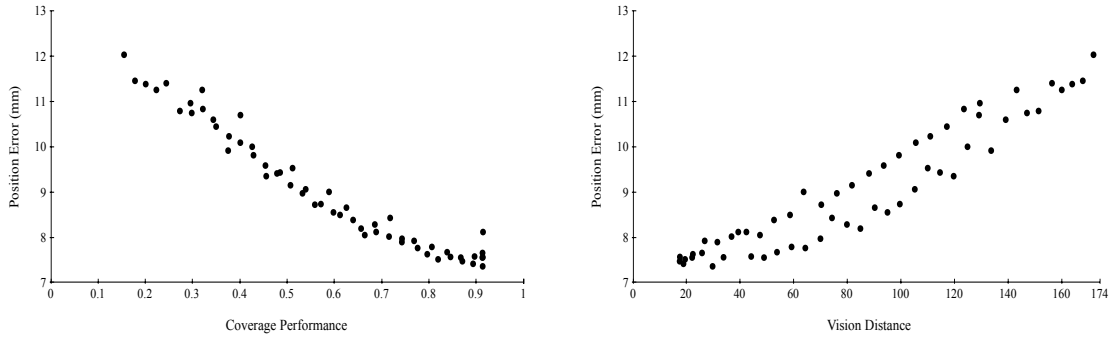
**Figure 5.2**: Comparison between the coverage strength model and the vision distance in their response to translational and rotational movements between a camera and a task.

From the results it can be seen that the coverage model is "unaware" of these changes, whereas the vision distance is able to report them.

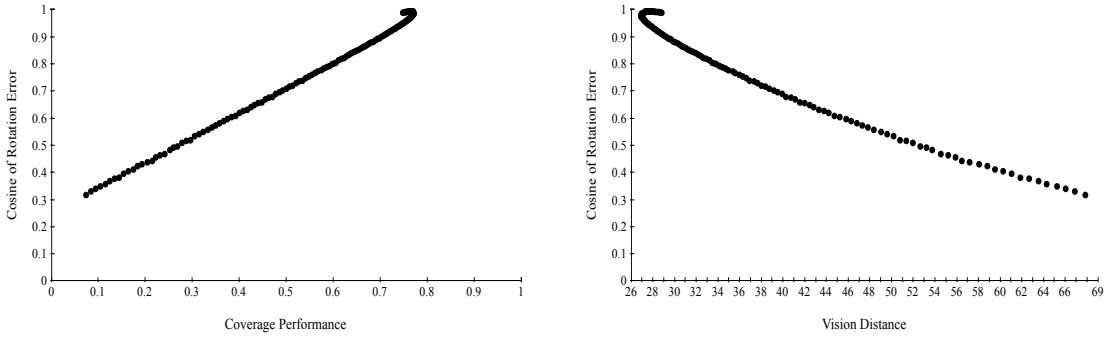
In this simulation both methods performed well. The coverage model and the vision distance were able to identify the optimal configuration between the camera and the task. The optimal configuration occurred when the task was located at the centre of the frustum and its normal aligned with the optical

axis.

### 5.3.2 Experiments




(a) CSM (b) VD




(c) CSM (d) VD

**Figure 5.3:** Correlation between the tracking error and the performance metric of the coverage strength model and the vision distance.

The vision distance was also tested using a physical apparatus (see Figure 5.4). In this experiment, a target of known geometry was placed on the end effector of a robotic manipulator. A camera was placed in front of the target and a tracking application was used to estimate the three-dimensional pose of the target. The robot's encoders also provided an estimate of the target's pose, which was used as the ground truth. The difference between the ground truth and the camera's estimate is the positioning error, which is measured in mm for translation, and as the cosine of the angular difference for rotation.

Similar to the simulation, the experiment was conducted twice and the response of both models is reported. During the first execution, only translational movement of the robotic arm along the optical axis was allowed (Figures 5.3(a) and 5.3(b)). During the second run, only rotational movement was allowed (see Figures 5.3(c) and 5.3(d)).

Figure 5.3 shows the comparison between the models' responses and the tracking error, and Table 5.3.2 shows the Pearson product-moment correlation coefficient for each of the figures. In this case the coverage strength model proved to be more accurate in its prediction of the vision system's performance. Although the vision distance's accuracy is very close to that of the coverage model.

**Figure 5.4**: Experimental Apparatus.

Recall that the vision distance approximates the viewing frustum with an ellipsoid, and thus it is not as accurate. However, the vision distance has proven to behave more linearly and thus it is a better candidate for optimization.

**Table 5.1**: Correlation Comparison

| Model | Translation | Rotation |
|---|---|---|
| CSM | -0.9715 | 0.9983 |
| VD | 0.9476 | -0.9858 |

# Part III

# Optimization

CHAPTER 6

# Deployment of Camera Networks for Industrial Inspections

Assure me of nothing! We assure ourselves.

Taraza, *Heretics of Dune* (1984)

## 6.1 Overview

The optimization of the camera parameters for a given task can be expressed in the following way: A camera is said to have an optimal position when the task lies at the centre of the viewing frustum. A camera is said to have an optimal orientation when the optical axis is collinear with the normal to the plane tangent to the surface of the task. In the case where the objective is to optimize a single camera and the task is composed of a single triangle, the optimal solution is found when the vision distance between the camera and the task is 0. However, in the case where the objective is to optimize multiple cameras and the task is composed of multiple triangles, finding a solution is not trivial. In Section 6.2 and Section 6.3 it is shown how to find an optimal solution when the problem involves a single camera and a task composed of multiple triangles. Finally, in Section 6.4 it is shown how to compute a suboptimal solution for multiple cameras.

Minimizing the vision distance between a camera and a set of triangles can be simplified by performing optimization separately in the group of rotations $SO(3)$ and in Euclidean space $\mathbb{R}^3$. This is possible because the special Euclidean group $SE(3)$ is homeomorphic to the product $SO(3) \times \mathbb{R}^3$. In short, the vision distance can be minimized by minimizing the Frobenius and Euclidean distances separately.

---

6 **Proposition ($SE(3)$ is Homeomorphic to $SO(3) \times \mathbb{R}^3$)**
*The special Euclidean group $SE(3)$ is the semidirect product of the special orthogonal group $SO(3)$ and the Euclidean space $\mathbb{R}^3$.*

Proof Any element in $SE(3)$ is a translation followed by a rotation

$$\mathbf{x} \mapsto A(\mathbf{x} + \mathbf{t}) \tag{6.1}$$

or a rotation followed by a translation

$$\mathbf{x} \mapsto A\mathbf{x} + \mathbf{t} \tag{6.2}$$

where $A$ is an orthonormal matrix, and $\mathbf{x}$ and $\mathbf{t}$ are elements in $\mathbb{R}^3$.

The translational group $T(3)$ and the special orthogonal group $SO(3)$ have elements in $\mathbb{R}^3$ and $\mathbb{R}^{3\times3}$, respectively, and both are subgroups of $SE(3)$. Since the translation and rotation operations are linear transformations, and rigid motions (6.1) and (6.2) in $SE(3)$ are also linear transformations; it follows that $SE(3)$ is homomorphic to the product $SO(3) \times \mathbb{R}^3$. ■

---

The previous proof implies that any element in $SE(3)$ can be decomposed into elements of $SO(3)$ and $\mathbb{R}^3$. Additionally, if all frames of reference "agree" upon the same convention for rigid body motions (e.g. either (6.1) or (6.2)), then the individual elements in $SE(3)$ can be treated separately (see Ma et al. [90]).

## 6.2 Single Camera Position Optimization

Given a list of $n$ triangles $[T^1, ..., T^n]$ with known positions $\mathbf{p}_T^k$, $k \in [1, n]$. The objective is to minimize the sum of squared differences between the camera $\mathbf{p}_C$ and every triangle $\mathbf{p}_T^k$. In order to ensure visibility, the optimization must be constrained so that the maximum distance between the camera and any triangle falls within a predetermined threshold. The threshold is chosen to be the geometric norm (4.7) of the camera tensor $C$. The choice of the threshold ensures that all the triangles will be inside the viewing frustum.

The optimization problem is formulated as

$$\text{Minimize} \quad \sqrt{\sum_{i=1}^{3} (p_{C_i} - p_{Ti}^k)^2} \tag{6.3a}$$

$$\text{subject to} \quad ||\mathbf{p}_C - \mathbf{p}_T^k||_2 \leq ||C||_G \tag{6.3b}$$

$$\text{for} \quad k = 1, 2, ..., n \tag{6.3c}$$

where $\mathbf{p}_C$ is the variable to be optimized, and $||C||_G$ is a known threshold.

The problem in (6.3a) to (6.3c) can be reformulated as a second-order cone programming problem by introducing a new variable $a$, which represents the upper bound of (6.3a). Additionally, a new vector $\mathbf{v}$ is introduced such that

$$||\mathbf{v}||_2 = \left\| \begin{array}{c} p_{C1} - p_{T1}^1 \\ p_{C2} - p_{T2}^1 \\ p_{C3} - p_{T3}^1 \\ \vdots \\ p_{C1} - p_{T1}^n \\ p_{C2} - p_{T2}^n \\ p_{C3} - p_{T3}^n \end{array} \right\|_2 = \sqrt{\sum_{i=1}^{3} (p_{C_i} - p_{Ti}^k)^2} \tag{6.4}$$

With the new variables in place, the optimization problem is reformulated as a standard second-order cone programming problem:

$$\begin{aligned} \text{Minimize} \quad & a && \text{(6.5a)} \\ \text{subject to} \quad & ||\mathbf{v}||_2 < a && \text{(6.5b)} \\ & ||\mathbf{p}_C - \mathbf{p}_T^k||_2 \leq ||C||_G && \text{(6.5c)} \\ & \text{for} \quad k = 1, 2, ..., n && \text{(6.5d)} \end{aligned}$$

The formulation of the previous optimization problem into a standard form is very advantageous, because there are many tools that can be used to find a solution (see Sra et al. [91]).

## 6.3 Single Camera Orientation Optimization

Finding the orientation of the camera that minimizes the Frobenius distance to the orientations of the triangles is similar to finding the "average" of the rotations. Given the normalized tensors $C$ and $T^k$ representing the rotations of the camera and $n$ triangles, respectively for $k \in [1, n]$. The problem can be expressed as follows:

$$\begin{aligned} \text{Minimize} \quad & \sqrt{\sum_{i=1}^{3} \sum_{j=1}^{3} (c_{ij} - t_{ij}^k)^2} && \text{(6.6a)} \\ \text{subject to} \quad & ||\mathbf{c}_1||_2 = 1 && \text{(6.6b)} \\ & ||\mathbf{c}_2||_2 = 1 && \text{(6.6c)} \\ & \mathbf{c}_1 \cdot \mathbf{c}_2 = 0 && \text{(6.6d)} \\ & \text{for} \quad k = 1, 2, ..., n && \text{(6.6e)} \end{aligned}$$

where $C$ is the variable to be computed, $\mathbf{c}_1 \cdot \mathbf{c}_2$ denotes the inner product, and $\mathbf{c}_3 = \mathbf{c}_1 \times \mathbf{c}_2$.

The problem in (6.6a) to (6.6e) is not a convex problem because (6.6b) to (6.6d) are second-order equality constraints, and as such, the solution is very hard to find. While applying other methods

such as heuristics or non-linear optimization could solve the problem. This dissertation is committed to finding a solution that is computationally efficient and with some guarantee of optimality. This problem is solved by performing optimization of the orientation component in the tangent space of $SO(3)$.

### Optimization in the Tangent Space

Let $\mathcal{O}$ be a smooth manifold containing all possible rotations in $SO(3)$ with elements $R \in \mathbb{R}^{3\times3}$: $R^{-1} = R^T$, $\det(R) = 1$. The tangent space of $\mathcal{O}$ at $R$, denoted as $T_R\mathcal{O}$, is defined as the span of the tangent vectors for all the possible curves passing through $R$. The logarithmic map $\log(R)$: $\mathcal{O} \mapsto T_R\mathcal{O}$ maps $R \in \mathcal{O}$ to $\mathbf{r} \in T_R\mathcal{O}$. The exponential map is the inverse of the logarithmic map, and it is defined as $\exp = \log^{-1}$. The exponential map is a diffeomorphism between a sufficiently small neighbourhood of 0 in $T_R\mathcal{O}$, and a neighbourhood of $R$ in $\mathcal{O}$. In this case the tangent space of $SO(3)$ is the space of $3 \times 3$ skew-symmetric matrices $so(3)$.

The logarithmic function $\mathrm{Log}(\cdot)$ transforms a matrix in $SO(3)$ into a skew-symmetric matrix in $so(3)$. The resulting matrix in $so(3)$ can in turn be expressed as a vector according to the following procedure

$$\log(R) = \mathbf{r} = \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix} \in so(3)$$
$$\hat{\mathbf{r}} = [r_1,\ r_2,\ r_3]^T \tag{6.7a}$$

The well known Rodrigues formula [92] can be used to compute the exponential map in the following way.

$$\exp(\mathbf{r}) = I\cos||\hat{\mathbf{r}}|| + \mathbf{r}\sin||\hat{\mathbf{r}}|| + (1 - \cos||\hat{\mathbf{r}}||)\hat{\mathbf{r}} \otimes \hat{\mathbf{r}} \tag{6.8}$$

where $I$ is the identity matrix and $\hat{\mathbf{r}} \otimes \hat{\mathbf{r}}$ is the tensor product as defined in (4.8).

The logarithmic map can be computed as

$$\theta = \arccos\left(\frac{trace(R) - 1}{2}\right) \tag{6.9}$$

$$\log(R) = \begin{cases} 0 & \text{if } \theta = 0 \\ \frac{\theta}{2\sin\theta}(R - R^T) & \text{if } \theta \neq 0,\ \theta \in [-\pi, \pi] \end{cases} \tag{6.10}$$

Since $so(3)$ is isometric to $\mathbb{R}^3$, $\mathbf{r}$ can be interpreted as a vector in $\mathbb{R}^3$ (e.g. the vector $\hat{\mathbf{r}}$), and compute the average of rotations using the second-order cone program defined in (6.5a) to (6.5d). The main advantage of the optimization in the tangent space is that the problem becomes convex, and the optimal solution can be computed easily. The only consideration to have is that the threshold in (6.5c) needs to be redefined, since its physical meaning does not translate well into the tangent space.

By operating in the tangent space $so(3)$, the curvature of $SO(3)$ can be accounted for and avoid the approximation issues of using the Frobenius distance directly. However, since the exponential map is a diffeomorphism of a sufficiently small neighbourhood $0 \in so(3)$ and $R \in SO(3)$, the optimization in

the tangent space is also an approximation. Thus the program in (6.5a) to (6.5d) converges to the true "average" of rotations only when all the rotations are close enough to each other. In this dissertation it is assumed that the task model is smooth enough for this purpose. In other words, the tensors of neighbouring triangles $T_j$ of any triangle $T_i$ in the task model are at most $d_F(T_i, T_j) < d_F(T_i, -T_i)$. Algorithm 3 shows how to optimize the orientation.

**Algorithm 3** Orientation Optimization

**Input:** list of triangles $\mathcal{T}$
**Output:** $C$

$S = [\,]$
$M = [\,]$
**for** $T \in \mathcal{T}$ **do**
  $T = \frac{T}{||T||}$ using (4.6)
  $\mathbf{t} = \mathrm{Log}(T)$ using (6.10)
  $\hat{\mathbf{t}} \leftarrow \mathbf{t}$ using (6.7)
  $S \leftarrow \hat{\mathbf{t}}$
  $M \leftarrow ||\hat{\mathbf{t}}||_2$
**end for**
substitute $\hat{\mathbf{c}}$ for $\mathbf{p}_C$ and $\hat{\mathbf{t}}$ for $\mathbf{p}_T$ in (6.5a) to (6.5d)
substitute $||C||_G$ for $\sqrt{\max(M)}$ in (6.5c)
$\hat{\mathbf{c}} \leftarrow$ optimal orientation using $S$ in (6.5a) to (6.5d)
$\mathbf{c} \leftarrow \hat{\mathbf{c}}$ using (6.7)
$C = \exp(\mathbf{c})$ using (6.8)
**return** $C$

The threshold $||C||_G$ in (6.5c) represents the average size of the viewing frustum of the camera, thus making sure that the program in (6.5a) to (6.5d) computes a position such that all the triangles are inside the frustum. In Algorithm 3, however, the threshold must be replaced by some meaningful metric. This metric must be representative of the requirement to have the orientation of the camera as close as possible to the orientations of the triangles. The threshold in (6.5c) for Algorithm 3 is chosen, empirically, to be the squared root of the maximum Euclidean distance between the vectors in the tangent space $so(3)$.

Algorithm 3 begins by transforming all triangle tensors from $SO(3)$ into the tangent space via (6.10). The triangles are then transformed into Euclidean space using (6.7). By constructing a list of the triangles represented as vectors in the tangent space, and a list of their magnitudes; a suitable threshold is found and the convex program in (6.5a) to (6.5d) is used to find the average of rotations for all the elements in the list of tangent vectors. The optimal orientation is also a vector in the tangent space and it is converted to matrix form using (6.7). The final rotation matrix representing the optimal orientation for the camera is found by transforming the solution from the tangent space into $SO(3)$ via (6.8).

## 6.4 Multi-Camera Deployment

In Chapter 3, a method was presented for deployment of multiple cameras using a graph-based heuristic approach. This method is used in this chapter to obtain an initial deployment of cameras, which

is then refined by the convex optimization presented in the previous sections.

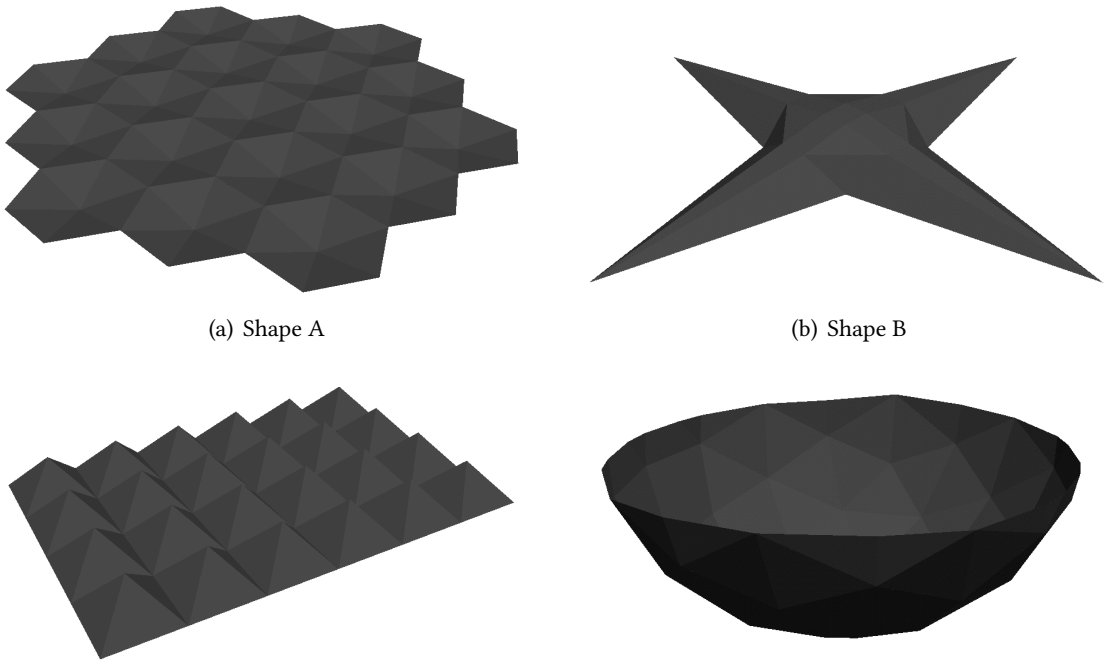
The initial deployment is performed as follows. The inputs are a triangular mesh model of the task (e.g. a CAD model), a list of task parameters (see Table 2.1), and a list of lenses and camera sensors to choose from. With the aforementioned inputs, the task model is discretized and a selection pool is populated. A viewpoint candidate is generated for optimal coverage[1] for every triangle in the task model. With the viewpoint candidates as the vertices, a graph of visual overlap is constructed. A greedy search selects viewpoints from the pool using some custom metric based on (2.1), and it removes those viewpoints found to be redundant in the overlap graph. A binary reach algorithm is used afterwards to further minimize the selection pool and produce the initial camera deployment, (see Chapter 3 for details).

The convex optimization program is used to refine the camera positions and orientations in the following way. For every camera position $\mathbf{p}_{Ci}$ in the initial deployment, every triangle $\mathbf{p}_{Tj}$ in the task model is checked and append to a list $l_i$ if it meets the condition $||\mathbf{p}_{Ci} - \mathbf{p}_{Tj}||_2 \leq ||C_i||_G$. An optimal camera position and orientation is obtained using the second-order cone program in (6.5a) to (6.5d) for the position, and using Algorithm 3 for the orientation. Algorithm 4 illustrates this process.

**Algorithm 4** Multi-Camera Deployment

**Input**: task model, task parameters, lens/sensor list.
**Output**: list of camera poses.

compute initial deployment as described in Chapter 3
with the initial deployment as a list of poses $\mathcal{P} \in SE(3)$, and a list of triangles $\mathcal{T}$
$L = [\,]$
**for** $C \in \mathcal{P}$ **do**
  $l = [\,]$
  **for** $T \in \mathcal{T}$ **do**
    **if** $||\mathbf{p}_C - \mathbf{p}_T||_2 \leq ||C||_G$ **then**
      $l \leftarrow T$
    **end if**
  **end for**
  $L \leftarrow l$
**end for**
$\hat{\mathcal{P}} = [\,]$
**for** $l \in L$ **do**
  $P \leftarrow$ optimal position using (6.5a) to (6.5d)
  $O \leftarrow$ optimal orientation using Algorithm 3
  $\hat{\mathcal{P}} \leftarrow$ Pose($P$,$O$)
**end for**
**return** $\hat{\mathcal{P}}$



(a) Shape A (b) Shape B

(c) Shape C (d) Shape D

**Figure 6.1**: Triangular mesh models of various tasks.

## 6.5 Experimental Validation

### 6.5.1 Simulations

In this experiment the convex optimization method is used to deploy a camera network for four different task models, which are shown in Figure 6.1. The camera networks are deployed and tested through simulation in Adolphus. The performance of the deployment is measured objectively using the coverage performance (2.1). The results are summarized in Table 6.1. The columns are described as follows. The second column shows the number of triangles in the task model. The third column shows the number of cameras selected applying the greedy search. The fourth column shows the number of cameras selected after the binary search. The fifth column shows the time needed to compute a solution, expressed in seconds. Finally, the last column shows the coverage performance as defined by (2.1).

**Table 6.1**: Performance of Convex Optimization Method

| Shape | Triangles | Greedy | Binary | Time | Performance |
|---|---|---|---|---|---|
| A | 114 | 6 | 6 | 8.09 | 0.9474 |
| B | 54 | 7 | 6 | 1.40 | 0.9687 |
| C | 96 | 10 | 8 | 5.30 | 0.9791 |
| D | 88 | 6 | 6 | 4.92 | 0.7951 |

[1]This process is trivial since it involves a single viewpoint and a single triangle.

### Comparisons

**Table 6.2**: Performance of Heuristic Method

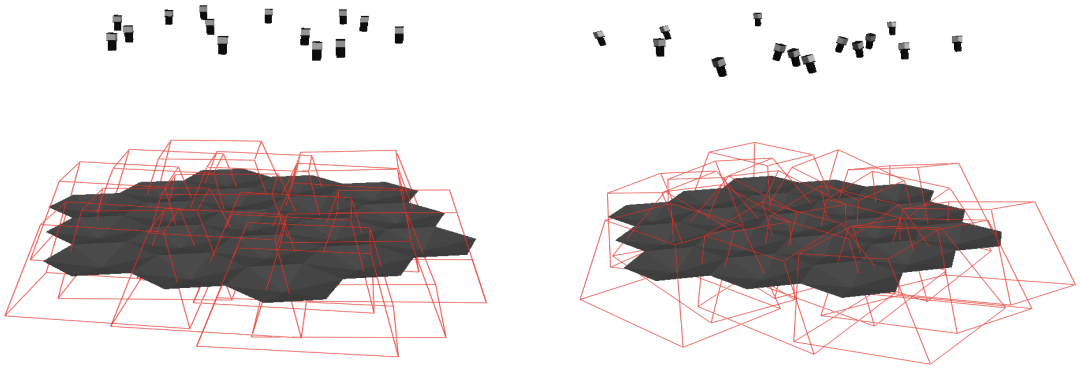
| Shape | Triangles | Greedy | Binary | Time | Performance |
|---|---|---|---|---|---|
| A | 114 | 6 | 6 | 8.27 | 0.8558 |
| B | 54 | 7 | 6 | 1.17 | 0.8899 |
| C | 96 | 10 | 8 | 4.58 | 0.8683 |
| D | 88 | 6 | 6 | 4.68 | 0.7386 |

The performance of the heuristic deployment method is also shown for the same tasks in the previous simulations. No convex optimization is used in this case. The results of the camera deployment can be seen in Table 6.2. In this experiment the time needed to compute a solution is less than in the previous experiment; however, the coverage performance is also less. Table 6.3 shows a comparison of the coverage performance between the two methods. A visual comparison of the two methods can be seen in Figure 6.2. In this case (shape A) the convex method achieves a more uniform deployment than the heuristic approach.

**Table 6.3**: Comparison between Heuristics and Convex Methods

| Method | Shape A | Shape B | Shape C | Shape D |
|---|---|---|---|---|
| Convex | 0.9474 | 0.9687 | 0.9791 | 0.7951 |
| Heuristic | 0.8558 | 0.8899 | 0.8683 | 0.7386 |
| **Difference** | 9.16% | 7.88% | 11.08% | 5.65% |

The results show that the deployment method based on convex optimization outperforms the heuristic method. The deployment method based on convex optimization refines the initial camera deployment by finding an optimal position and orientation for each of the cameras in the initial solution.



(a) Convex Optimization (b) Heuristic Method

**Figure 6.2**: Multi-Camera Deployment.

### 6.5.2 Experiments

In what follows the camera deployment obtained using the convex method is implemented. The performance of the camera network is evaluated using shape C from Figure 6.1. The physical apparatus includes a cardboard model of shape C at a 1-to-1 scale. The dimensions of the model are 960mm × 640mm × 80mm.

The camera network is implemented by first defining a global reference frame (grf) in the workspace. The cameras are placed manually by installing them at their corresponding positions and orientations. In order to reduce the positioning error, the cameras are adjusted as necessary using camera calibration as feedback. The cameras are externally calibrated by aligning a calibration target to the grf (see Figure 6.3).

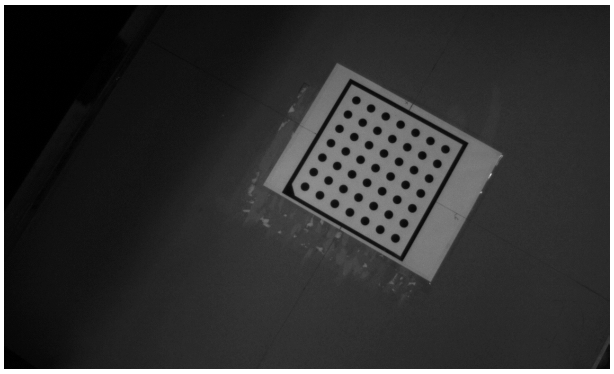


**Figure 6.3**: Calibration Plate

With the position and orientation of the cameras as measured using calibration, the positioning error is defined as follows

$$e_p = \max(\mathbf{p}_{Ci}^{\text{act}} - \mathbf{p}_{Ci}^{\text{exp}}) \quad i = 1, 2, 3. \tag{6.11}$$

where the superscript "act" stands for actual, and "exp" stands for expected. $\mathbf{p}_{Ci}$ is the i$^{\text{th}}$ component of the camera's position.

The orientation error is defined similarly

$$e_o = \max(\hat{\mathbf{r}}_i^{\text{act}} - \hat{\mathbf{r}}_i^{\text{exp}}) \quad i = 1, 2, 3. \tag{6.12}$$

where $\hat{\mathbf{r}}_i$ is the i$^{\text{th}}$ component of the vector in the tangent space of the camera's rotation matrix (see (6.7)).

The cameras are adjusted and calibrated repeatedly until the positioning and orientation erros fall below some acceptable threshold. The final implementation can be seen in Figure 6.4. The positioning error can be visualized in Figure 6.5 where a simulation was created using the camera positions and orientations measured from calibration. In Figure 6.5 the opaque cameras represent the cameras installed in the physical apparatus, and the pale cameras represent the positions as expected in the original camera deployment.

The performance of the camera network was evaluated by measuring the effective area of the cardboard model that is visible to the cameras. By taking an image from each camera in the apparatus,

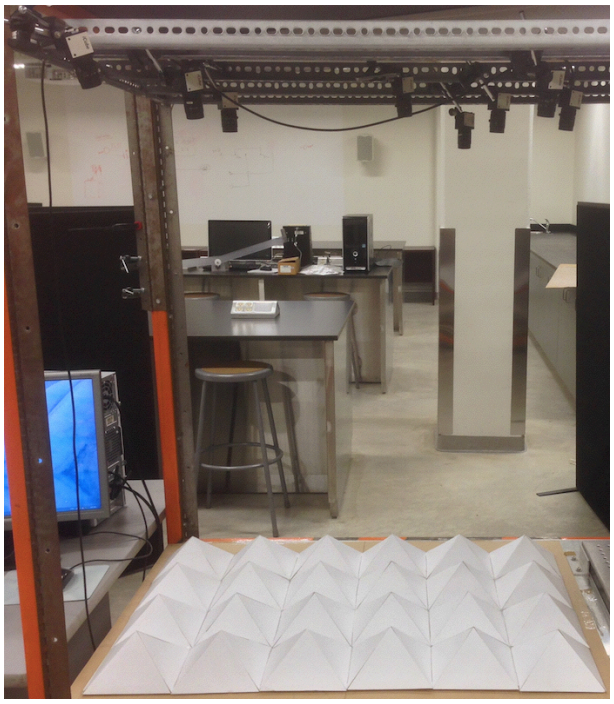


**Figure 6.4**: Experimental Apparatus - Deployment using convex method.

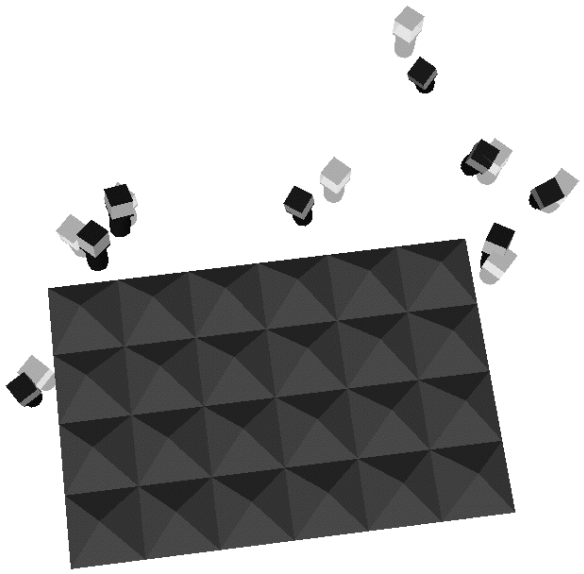


**Figure 6.5**: Positioning Error - Opaque cameras represent the actual positions as illustrated in Figure 6.4

a series of basic image processing steps were taken to segment the triangles of the cardboard model. The area of each triangle was calculated in pixels using blob analysis. The total visible area was calculated by simply adding the area of all the triangles and by normalizing the result using the

maximum area. The maximum area was obtained by measuring the area in pixels of a single non-skewed triangle and by multiplying its area by the total number of triangles in the cardboard model. The threshold for the positioning error is $e_p = 70$mm, and the threshold for the orientation error is $e_o = 0.3$rad. The performance of the camera network in Figure 6.4 is quantified in Table 6.4.

**Table 6.4**: Performance of camera network in Figure 6.4

| $e_p$ | $e_o$ | **Visible Area** |
|---|---|---|
| 62.501mm | 0.23rad | 92.34% |

### Comparisons

A method for deployment of camera networks based on particle swarm optimization is implemented, based on [38, 93, 94], in order to compare it to the method presented in this chapter. The implementation and the details about the particle swarm method are described in more detail in Chapter 7.

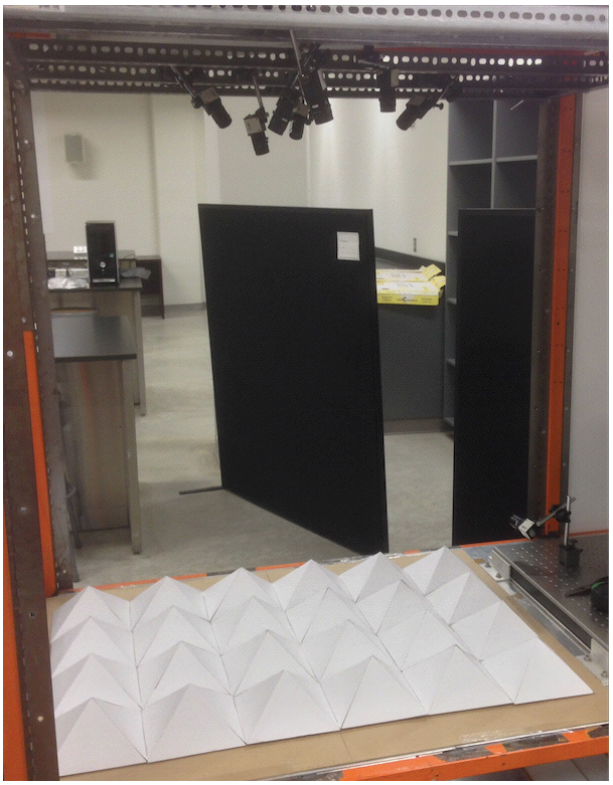


**Figure 6.6**: Experimental Apparatus - Deployment using the particle swarm method.

The experiment using the camera deployment produced by the particle swarm method is executed following the same steps as in the previous evaluation. The physical implementation is shown in Figure 6.6, and the error visualization is shown in Figure 6.7. In this experiment the same error thresholds where used; $e_p = 70$mm and $e_o = 0.3$rad.

The performance of this camera deployment was measured using the same procedures as in the previous evaluation and the results can be seen in Table 6.5. By comparing Tables 6.4 and 6.5 the

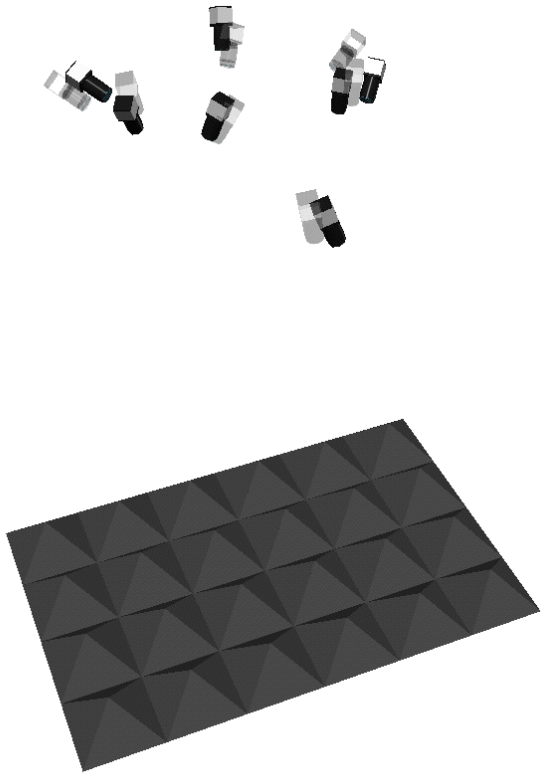


**Figure 6.7**: Positioning Error - Opaque cameras represent the actual positions in Figure 6.6

conclusion can be made that the convex method presented in this chapter outperforms the particle swarm method in the deployment of camera networks.

**Table 6.5**: Performance of camera network in Figure 6.6

| $e_p$ | $e_o$ | **Visible Area** |
|---|---|---|
| 49.54mm | 0.08rad | 82.12% |

### 6.5.3 Robustness

While the implementation of the camera deployment provided by both methods incurred in some error. The error in the implementation of the deployment provided by the convex method is much larger than the error found in the deployment provided by the particle swarm optimization method. However, the decrease in performance of the deployment using the convex method is relatively the same as that of the particle swarm method (see Table 6.6). Thus, the conclusion can be made that the convex method is more robust to positioning error than the particle swarm method.

An intuitive comparison can be made between the two methods in Figures 6.8 and 6.9. Figure 6.8 shows the view from a particular camera in the deployment method obtained using the convex method. The simulated view can be seen on the left, and the actual view can be seen on the right. Similarly, Figure 6.9 shows the same information about the deployment obtained using the particle

**Table 6.6**: Performance comparison

| **Method** | $e_p$ | $e_o$ | **Visible Area** | **Expected Performance** |
|---|---|---|---|---|
| CVX | 62.501mm | 0.23rad | 92.34% | 97.91% |
| PSO | 49.54mm | 0.08rad | 82.12% | 87.56% |
| **Difference** | 20.73% | 65.21% | 10.22% | 10.35% |

swarm method. From this comparison it is possible to see that the convex method performs a more efficient use of cameras, while the particle swarm method leaves some cameras with fields of view that are mostly unused.

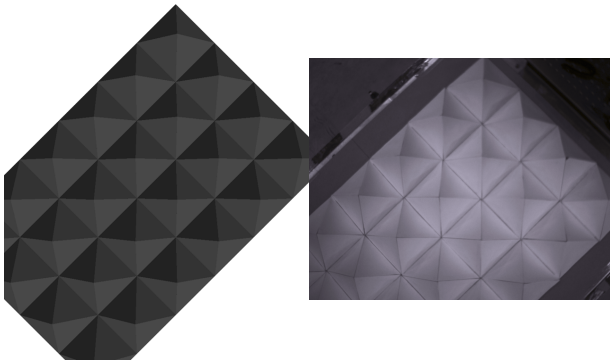


**Figure 6.8**: Simulation vs Reality - A comparison between the simulation and the physical implementation for the deployment obtained using the convex method.

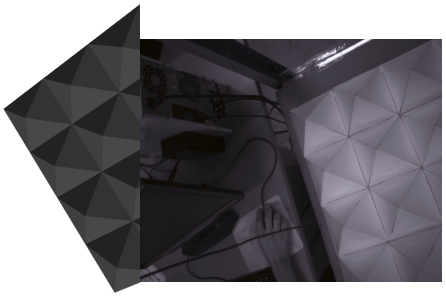


**Figure 6.9**: Simulation vs Reality - A comparison between the simulation and the physical implementation for the deployment obtained using the particle swarm method.

CHAPTER 7

# Multi-Camera Deployment for Surveillance Applications

> If you put away those who report accurately, you'll keep only those who know what you want to hear. I can think of nothing more poisonous than to rot in the stink of your own reflections.
>
> Jessica, *Children of Dune* (1976)

## 7.1 Overview

Deployment of camera networks has an important use in security applications, and chief among these is surveillance. In this section, a deployment method is developed by presenting a modified version of the deployment method in Chapter 6. The goal is to produce a multi-camera configuration that can function as a surveillance system. The main diferences between a surveillance system and an inspection system are the task and the task's requirements. In an inspection system the task is often represented by a CAD model of the object or target. Although a surveillance system also has the goal of imaging a target or targets (e.g. people and/or objects of interest), in this case the target is often unknown and dynamic.

The remainder of this section shows, in Section 7.2.1, how to construct a task such that the needs and requirements of a typical surveillance system are satisfied. Section 7.2.2 presents a modified version of the method in Section 3.3.2 that accounts for the unknown dynamics of the target or targets by oversampling the scene. Section 7.2.3 presents the deployment method and Section 7.3 makes a qualitative comparison with existing work. Finally, Section 7.4 presents the results and simulations of the proposed method, and compares it to the implementation of a method based on particle swarm optimization.

## 7.2 Deployment Method

The work in this section is limited to deployment of camera networks for buildings or areas where a CAD model of the floor plan is available. The task and the solution space are generated by oversampling the scene. Although no assumption is made about the location, number, or motion of the target or targets, this section assumes that the geometry of the scene is known a priori.

Initially, the CAD model of the scene is segmented into three different parts. First, the *floor* plan is separated from the rest of the model, and it is used to construct a suitable task. Second, the *walls* are separated and used to compute static occlusion. Third, a section of the walls is segmented and defined as the feasible area from wich viewpoints are generated. In practice, the third section is usually a cross section of top of the walls, and it represents the feasible areas in which cameras may be mounted. Thus, this third section is named *mounts*.

### 7.2.1 Task Definition

Section 3.3.1 takes advantage of the triangular mesh as a well defined task to generate an optimal viewpoint candidate for each triangle in the task. The difficulty in this case is that the task is not known to the deployment method. Thus, the method should produce a camera configuration that covers as much of the scene as possible.

In order to represent the scene, and to make a fair comparison later in Section 7.3, the task is constructed as a set of points. All points in the task are coplanar and each point is generated by taking the centre of each triangle in the floor plan. The task may be located at the same height as the floor. However, taking into account that a surveillance system may be more interested in the faces of human targets rather than their feet, the task may be translated upwards as defined by the user. The points in the task model account for all possible positions of the targets in the scene[1].

The target or targets may take on any position and orientation within the floor plan. The possible orientations are represented using directional points. For each triangle centre in the floor plan, an arbitrary number of directional points is added with direction componentes, $\rho$ and $\eta$, that are uniformly distributed in the ranges $[\frac{\pi}{2}, \frac{\pi}{2}]$ and $[0, 2\pi]$ respectively[2]. Note that the angle ranges and the number of directional points can be adjusted to balance the tradeoff between accuracy and the computational cost of finding a solution.

### 7.2.2 Solution Space

Due to practical considerations, cameras may only be installed atop of walls. The solution space is generated from the mounts in the following way. For each triangle in the mounts, a line of sight is defined between the triangle's centre and each of the points in the floor plan. Each line of sight represents a viewpoint candidate in the solution space. A viewpoint candidate must meet certain requirements before it can be considered as a feasible candidate.

1. The magnitude of the line of sight of a viewpoint may not be smaller than $z_n$, according to (2.6).

[1]The task models the scene to an arbitrary degree of precision, which can always be increased by performing an additional triangulation step on the floor plan.

[2]The direction components $\rho$ and $\eta$ are formated in spherical coordinates.

2. The magnitude of a viewpoint's line of sight must be equal or smaller than $z_f$, according to (2.7).

3. The magnitude of the line of sight of a viewpoint must not be greater than $z_R$, according to (2.8).

The previous conditions ensure that only those viewpoints that satisfy the focus, resolution, and visibility requirements of the task are considered. The reasoning is as follows. Firstly, By selecting viewpoints with lines of sight in the range $[z_n, z_f]$, this method ensures that every point in the task falls within the depth of field of at least one viewpoint. Secondly, any point closer than $z_R$ to any camera will be imaged with sufficient resolution. Finally, visibility is satisfied since every point in the task belongs to at least one line of sight in the list of viewpoint candidates.

Enforcing the previous conditions assumes that $z_f \geq z_R \geq z_n$. However, this assumption need not be enforced. The reader may refer to the optimization of the optical parameters for the selection of lenses and sensors in Section 3.4.

#### Static Occlusion

A viewpoint candidate may also be removed from the solution space if its line of sight is interrupted by an occluding surface in the scene. The walls represent such occluding surfaces and occlusion is determined in the following way. For every line of sight and for every triangle in the walls, if a line of sight is intersected by any triangle, the line of sight is removed from the solution space. Moller and Trumbore [95] offer a fast way of computing these intersections.

### 7.2.3 Multi-Camera Deployment

Using the task and solution space defined previously, an initial solution to the deployment problem is found by applying the method described in Section 3.5. The directional points in the task are used as the rows, and the viewpoint candidates are used as the columns of the vision matrix in Section 3.5.2. The initial deployment is obtained by applying Algorithm 1, for viewpoint selection, and Algorithm 2, for cost minimization.

Since a viewpoint candidate was generated as a line of sight between each of the triangles' centres in the mounts and each point in the task. This means that there are several viewpoints in the initial solution that have the same position but a different orientation. The final solution is found by treating the orientation of each viewpoint as a tensor. The method then replaces all the viewpoints that have the same position with a viewpoint that has an orientation equal to the average of the orientations of the viewpoints at that position. This process is repeated for every group of viewpoints with overlapping positions. Finally, the average of orientations is performed only for those tensors that are close enough to each other. This measure of closeness is represented as a threshold on the Frobenius distance between any two orientations (5.5). The optimal orientation of the viewpoints is found using the optimization in the tangent space described in Section 6.3. Algorithm 5 shows the refinement process.

**Algorithm 5** Multi-Camera Deployment for Surveillance

**Input**: Initial deployment $\hat{\mathcal{C}}$
**Output**: Final deployment $\mathcal{C}$

$\mathcal{C} = [\,]$
**while** $\hat{\mathcal{C}}$ **do**
  $pivot = \hat{\mathcal{C}}[0]$
  $temp = [\,]$
  $temp \leftarrow pivot.R$
  remove $pivot$ from $\hat{\mathcal{C}}$
  $\hat{\mathcal{C}}_c = \text{copy}(\hat{\mathcal{C}})$
  **for** $pose \in \hat{\mathcal{C}}_c$ **do**
    **if** $pose.T == pivot.T$ **then**
      $temp \leftarrow pose.R$
      remove $pose$ from $\hat{\mathcal{C}}$
    **end if**
  **end for**
  **if** $\text{len}(temp) == 1$ **then**
    $\mathcal{C} \leftarrow \text{Pose}(pivot.T, temp[0])$
  **else**
    **while** $temp$ **do**
      $t_a = \text{Tensor}(temp[0])$
      $temp_2 = [\,]$
      $temp_2 \leftarrow t_a$
      remove $t_a$ from $temp$
      $temp_c = \text{copy}(temp)$
      **for** $R \in temp_c$ **do**
        $t_b = \text{Tensor}(R)$
        **if** $d_F(t_a, t_b) < \frac{\pi}{4}$ **then**
          $temp2 \leftarrow R$
          remove $R$ from $temp$
        **end if**
      **end for**
      $R \leftarrow$ optimal orientation using Algorithm 3 on $temp$
      $\mathcal{C} \leftarrow \text{Pose}(pivot.T, R)$
    **end while**
  **end if**
**end while**
**return** $\mathcal{C}$

## 7.3 Comparison with Existing Work

The literature is rich with similar work that offers methods for deployment of camera networks within the context of surveillance applications. Recently, Reddy and Conci [38] presented one such method and used particle swarm optimization in order to compute a feasible solution.

In [38] the authors proposed a method that finds the positions and orientations that maximize visual coverage of a floor plan. In what follows, a qualitative comparison is made between the method

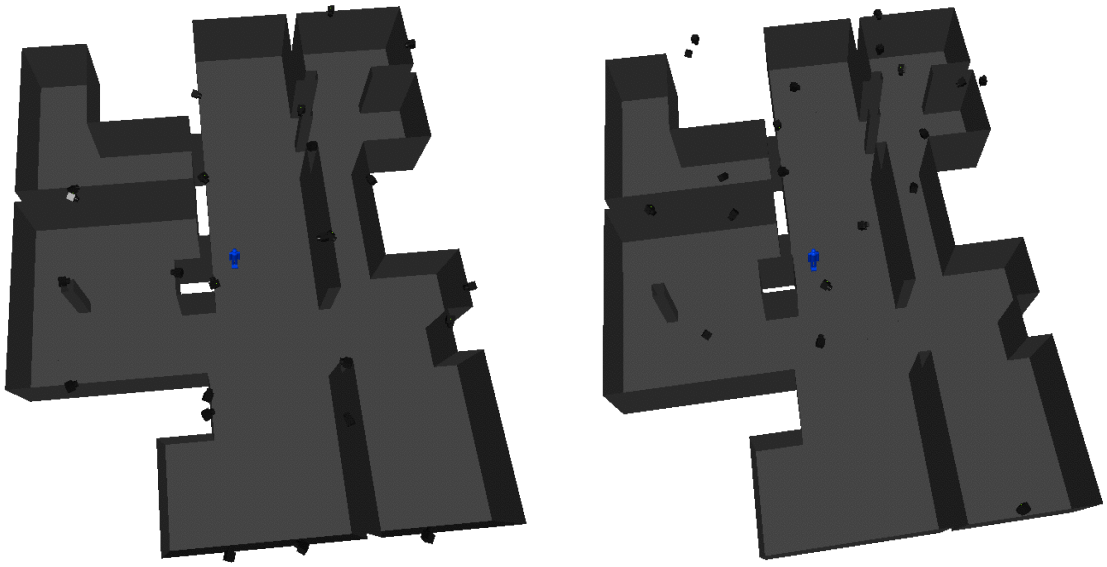
in this chapter and the method presented in [38].

- *Dimensionality.* The optimization in [38] is performed in two dimensions using two variables for position and one for orientation. The dimensionality of the method presented in this chapter can be regarded as being 2.5. Although the solution space uses three variables for position and two for orientation, all the points in the task are restricted to a plane that is parallel to the floor plan, thus the method is not entirely three-dimensional.
- *Inputs.* The work in [38] takes as an input an image, in raster format, of the floor plan where the camera network is to be deployed. The method in this chapter uses a CAD model of the floor plan, which includes the walls in order to compute occlusion.
- *Task Definition.* The method in this chapter samples the floor plan and places a point of interest at the centre of each triangle in the CAD model of the floor plan. The floor plan may be undersampled or oversampled to increase accuracy. The work in [38] asumes the number of targets is known a priori and places them randomly over the floor plan.
- *Optics.* The deployment method may, additionally, find the appropriate configuration of lenses and sensors. This type of optimization is addressed in this dissertation in Section 3.4. In [38], however the optics are assumed to be known a priori.
- *Cost Minimization.* Since the number of targets is assumed to be known a priori, Reddy and Conci assume that a number of cameras equal to the number of targets is sufficient. The authors do in fact provide study cases where the number of cameras is set to be either less or greater than the number of targets. On the other hand, the work in this chapter describes a way to automatically minimize the number of cameras from a richly populated solution space.
- *Illumination.* Reddy and Conci incorporate illumination in their camera model and thus produce camera configurations that account for the various lighting conditions in the scene. The work in this chapter assumes that illumination is optimal in every case and thus is not considered.
- *Optimization Method.* The main difference between the two methods is the optimization approach. The work in this chapter uses a greedy search based on graph tools for viewpoint selection and a binary search for cost reduction, followed by an optimization of the orientation of the viewpoints using convex programming. The method proposed in [38] uses particle swarm optimization. Where particle swarm optimization is a randomized process, convex optimization is known to be optimal. Thus, the method in this chapter offers at least a guarantee of local optimality.

## 7.4 Validation and Comparison

A method for deployment of camera networks based on particle swarm optimization is implemented in order to compare it to the method presented in this chapter. Since Reddy and Conci do not provide enough details about their implementation, the implementation in this work defaults to the work in [93, 94].

In this comparison both methods are subjected to the same input. The method using particle swarm optimization differs in that the solution space is simply defined as the limits in which its variables are allowed to change. Since the solution space is parametrized by $x$ and $y$ for position, and $\rho$ and $\eta$ for orientation. The solution space is defined by finding the bounds with in the floor plan. Finally, the fitness function is defined to be the coverage performance $F$ in (2.1).



(a) Convex Optimization (b) Particle Swarm Optimization

**Figure 7.1**: Result and simulation of the camera deployment methods.

The camera network obtained by each method is shown in Figure 7.1. Due to the large computational cost of constraining the viewpoints to walls in the fitness function. The particle swarm optimization method was allowed to place cameras on the ceiling as well as the walls.

Table 7.1 shows the number of iterations that where needed to reach the coverage performance of 0.2990 in the particle swarm optimization. The method was setup to deploy 25 cameras and it required 55 minutes to reach a stopping point. In this case, the deployment method stopped when the algorithm kept producing the same result over a series of iterations.

**Table 7.1**: Deployment Results PSO

| Iterations | Cameras | Time | Performance |
|---|---|---|---|
| 200 | 25 | 55.0 min | 0.2990 |

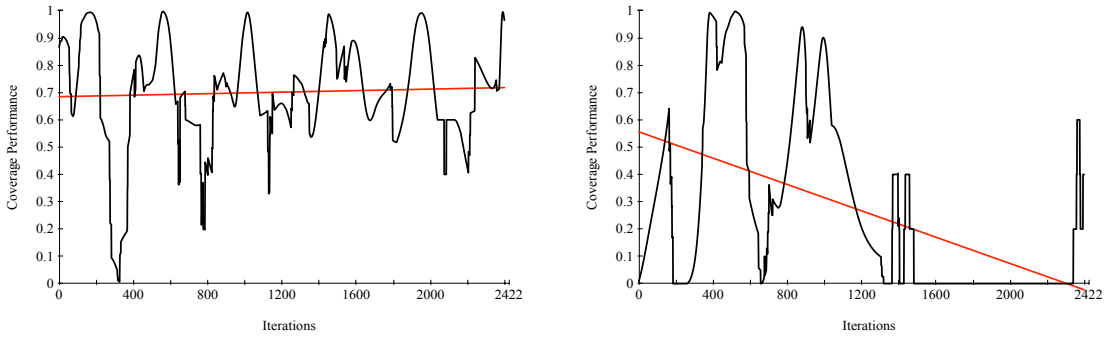
Table 7.2 shows the results of the method presented in this chapter. The solution space consisted of 262 viewpoints. The final solution was optimized down to 25 cameras in 0.35 minutes, while achieving a coverage performance of 0.3613.

In both tables the coverage performance is measured with respect to the task constructed as described in Section 7.2.1. In what follows, this task is discarded and replaced by a task that models a human target. A simulation is performed in which the human target moves around the scene and

**Table 7.2**: Deployment Results CVX

| Size | Cameras | Time | Performance |
|---|---|---|---|
| 262 | 25 | 0.35 min | 0.3613 |

the coverage performance provided by each camera netwrok is recorded. The results can be seen in Figure 7.2. The average performance in Figure 7.2(a) is 0.7015, and 0.2668 for Figure 7.2(b).




(a) Convex Optimization

(b) Particle Swarm Optimization

**Figure 7.2**: Comparison of the performance when tracking a human target between the camera deployments produced by different methods.

The results in Tables 7.1 and 7.2, and Figure 7.2 indicate that the method presented in this theis outperforms the one based on particle swarm optimization in terms of time and performance.

# Conclusions

# CHAPTER 8

# Conclusions

There are no facts, only interpretations.

Friedrich Nietzsche (1844–1900)

## 8.1 Summary of Contributions

An initial minor contribution is found in Chapter 3, where a method for deployment of multiple cameras based on graph tools and a heuristic implementation is presented. The method is formulated as a design tool that only needs as inputs a model of the task and a sensor-lens list. The output is a suboptimal solution to the multi-camera deployment problem. The heuristic method is validated using simulations and experiments in Section 3.6.

This dissertation presents two major contributions. Part II presents a novel framework for modeling the relationship between a camera and the task as a distance function. This distance, termed the *vision distance*, is based on tensors and a geometrical interpretation of the vision system. The vision distance effectively measures the distance between two visual entities, where a visual entity may be a camera or a task. The distance function measures not only the distance in the Euclidean sense, but also the distance between orientations in the Frobenius sense.

The advantages of the proposed approach are the following. The vision distance is a simple and easy-to-use metric that can be used to measure the performance of the vision system for a given task. The vision distance also exhibits some desirable properties such as the reduction to the Euclidean distance when the Frobenius distance is 0. Furthermore, the uniqueness of the global and local optimum under certain circumstances is showcased in Section 5.3.1. Simulations and experiments are used in Section 5.3 to validate the predictive capacity of the vision distance when measuring the performance of the vision system.

One of the most important advantages is that the framework in this dissertation allows the formulation of a convex optimization approach from a complex model of the vision system. Part III presents a deployment method for multi-camera networks based on convex optimization using second-order cone programming. Section 6.2 shows how to optimize the position of a camera for maximum coverage of a set of triangles. Additionally, Section 6.3 shows how to transform the orientation problem

into its tangent space, thus making the problem easy to be solved by the same convex approach. Finally, Section 6.4 combines the heuristic method presented in Chapter 3 with the convex optimization method. The result is a method that computes an initial deployment of multiple cameras, which is suboptimal. The initial solution is further optimized by the methods presented in Section 6.2 and Section 6.3. Section 6.5 presents simulations, experiments, and comparisons as a mean to validate the effectiveness of the deployment method presented in this dissertation.

Chapter 7 presents an application case of the optimization framework presented in this dissertation. A method for automatic deployment of camera networks is presented in Section 7.2, with special considerations for surveillance systems. Section 7.3 presents a qualitative comparison with similar existing work, that is based on particle swarm optimization. Finally, Section 7.4 offers a quantitative comparison between the two methods.

## 8.2 Conclusions

The tensor framework and the vision distance resulted in a sufficiently accurate model of the vision system. Experiments and comparisons with previous work showed that the decrease in accuracy from the original coverage model was small enough. When comparing with the gains in the ease of optimization, the vision distance emerged as the better choice for optimization applications.

The vision distance presented in this dissertation proved to be very useful in the formulation of two optimization approaches for multi-camera deployment. A combination of convex optimization and heuristics for local optimality and global sub-optimality, respectively, was presented with applications to industrial inspections and surveillance. These applications performed well when evaluated using the coverage model's performance index. Additionally, when compared to an implementation of camera-deployment using particle swarm optimization, The methods presented in this dissertation proved to be faster and performed better.

## 8.3 Directions for Future Work

The framework presented in this dissertation, which uses a tensor to describe some physical properties of the vision system in a unified manner, owes much of its merit to the field of differential geometry. This framework lays out a way to represent complex concepts such as the viewing frustum in a simple and compact manner; allowing a variety of applications such as modeling the vision system or enabling efficient optimization methods. In particular, optimization in the tangent space was inspired greatly from a review of other treaties in differential geometry.

This dissertation proposes a solution to the problem of multi-camera deployment. One of the assumptions made is that there is a model of the task, which is available as a triangular mesh. This, however, is not always the case and in some cases the task model is only available as a point cloud. Although Chapter 3 hinted at the possibility of handling point clouds by performing a triangulation or tessellation operation; this dissertation does not include any such cases as they fall outside of the scope.

Recovering the topological structure from a point cloud is an engaged endeavour on its own [96, 97, 98]. This problem is often approached by parametrizing the point cloud using an extensive

tool set from differential and Riemannian geometry. This work has applications in computational geometry where point cloud meshing is used to perform data fusion [99, 100]. However, in the field of computer vision this type of work can be used to classify tasks. Tasks can be classified based not only on their visual requirements but also on their shape. Tung and Matsuyama [68] touch the core of this problem by defining a global geodesic distance for 3D manifold meshes. In the case of triangular meshes, Chapter 6 may offer some possibilities; a local smoothens criterion may be defined, in the tangent space, to classify shapes. The field of computer vision could benefit greatly from such work; a classification framework could be used to identify the optimization approaches of camera networks that best suit the different types of tasks.

# Part IV

# Appendices

# APPENDIX A

# Mathematical Background

## A.1 Differential Geometry

This section briefly presents several concepts of differential geometry, which are used in this dissertation. Manifolds and differentiable manifolds are presented here for completeness and as a prerequisite for the definition of the tangent space.

### A.1.1 Manifolds

A manifold $\mathcal{M}$ is an $n$-dimensional topological space where each point $\mathbf{p} \in \mathcal{M}$ has a neighbourhood that is homeomorphic to the Euclidean space of dimension $n$.

A manifold is a generalization of the Euclidean space in that the underlying space need not be "straight" or "flat". Examples of one-dimensional manifolds are lines and curves, whereas examples of two-dimensional manifolds are planes and spheres.

While manifolds allow more complicated structures to be described, often they do not come equipped with a distance function, and thus are not metric spaces by default.

### A.1.2 Differentiable Manifolds

In geometry and topology, all manifolds are topological manifolds, with the possible inclusion of an additional structure such as a differentiable one. A differentiable manifold $\mathcal{M}$ is a type of manifold where every local neighbourhood around $\mathbf{p} \in \mathcal{M}$ is close enough to a linear space, thus enabling calculus to be performed.

Given an $n$-dimensional differentiable manifold $\mathcal{M}$, each point $\mathbf{p} \in \mathcal{M}$ has a tangent space. This is an $n$-dimensional Euclidean space consisting of the tangent vectors of the curves through $\mathbf{p}$.

### A.1.3 Tangent Space

Given a differentiable manifold $\mathcal{M} \in \mathbb{R}^n$, for every point $\mathbf{p} \in \mathcal{M}$ there exists a tangent space. The tangent space is a real vector space that contains all the possible "directions" that tangentially pass through $\mathbf{p}$. The elements of the tangent space are called tangent vectors at $\mathbf{p}$.

Let $\mathbf{x} = [x^1, ..., x^n]$ be Euclidean coordinates of $\mathbb{R}^n$, $\mathbf{p} \in \mathcal{M}$. The tangent space of $\mathcal{M}$ at the point $\mathbf{p}$,

$$T_{\mathbf{p}}\mathcal{M}$$

is the tangent space $\{\mathbf{p}\} \times E$, where $E$ is the $n$-dimensional vector space spanned by the basis

$$\frac{\partial}{\partial x^1}, ..., \frac{\partial}{\partial x^n}$$

which are the partial derivatives at point $\mathbf{p}$.

## A.2 Motion of Rigid Bodies

The Euclidean group $E(n)$ is a subgroup of the group of affine transformations. It has as subgroups the translational group $T(n)$, and the orthogonal group $O(n)$. $T(n)$ and $O(n)$ are groups of linear transformations in $\mathbb{R}^n$, these transformations define the translation and rotation operations, respectively.

### A.2.1 Translation

A translation is a mapping $T : \mathbb{R}^n \mapsto \mathbb{R}^n$ that moves every point $\mathbf{p} \in \mathbb{R}^n$ a constant distance in a specified direction. Given points $\mathbf{p}, \mathbf{v} \in \mathbb{R}^3$, translation can be defined in vector form as

$$T_{\mathbf{v}}\mathbf{p} = \mathbf{p} + \mathbf{v} \tag{A.1}$$

or in matrix form as

$$T_{\mathbf{v}}\mathbf{p} = \begin{bmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \mathbf{p} + \mathbf{v} \tag{A.2}$$

which is expressed in homogeneous coordinates.

### A.2.2 Rotation

A rotation is a mapping $R : \mathbb{R}^n \mapsto \mathbb{R}^n$ of the group of distance-preserving transformations. A rotation is a linear transformation that preserves the notions of distances and angles. This mapping transforms one orthonormal basis of $\mathbb{R}^n$ into another and it can be expressed as a matrix. The collection of orthogonal matrices $\mathbb{R}^{n \times n}$ with determinant equal to 1 or $-1$ is the orthogonal group $O(n)$, and the collection of orthogonal matrices with determinant equal to 1 is the special orthogonal group $SO(n)$.

A rotation matrix $R \in SO(3) : \det(R) = 1, R^T = R^{-1}$ defines a linear transformation in matrix form as

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{A.3}$$

which can be used to define a rotation operation for a point $\mathbf{p} \in \mathbb{R}^3$ using homogeneous coordinates as

$$R\mathbf{p} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} \tag{A.4}$$

### A.2.3 Pose

A pose is a linear mapping $P : \mathbb{R}^n \mapsto \mathbb{R}^n$ also known as an isometry because it preserves distances and angles for all $\mathbf{p} \in \mathbb{R}^n$. A pose combines the translation and rotation operations into a transformation known as a rigid motion. The collection of such transformations under $\mathbb{R}^3$ is known as the special Euclidean group $SE(3)$. Any element in $SE(3)$ is a translation followed by a rotation

$$\mathbf{p} \mapsto R(\mathbf{p} + \mathbf{v}) \tag{A.5}$$

or a rotation followed by a translation

$$\mathbf{p} \mapsto R\mathbf{p} + \mathbf{v} \tag{A.6}$$

This dissertation adopts the later as the standard convention for rigid motions. A transformation in $SE(3)$ can be expressed in matrix form using homogeneous coordinates as

$$P(\mathbf{p}) = \begin{bmatrix} r_{11} & r_{12} & r_{13} & v_x \\ r_{21} & r_{22} & r_{23} & v_y \\ r_{31} & r_{32} & r_{33} & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} \tag{A.7}$$

APPENDIX B

# Equipment Specifications

## B.1 Cameras

There are a few camera models available at the Graduate Control and Robotics Laboratory. Those used in the experiments presented in this dissertation are shown here with specifications. In all cases the cameras are used for monochrome imaging only. When the bayer filter for color imaging is available the settings are simply forced to monochrome.

### B.1.1 NET iCube NS4133BU

The NET iCube is a compact form factor monochrome CMOS camera based on the USB 2.0 standard. It has a resolution of 1.3 megapixels. This camera is 33mm × 33mm × 33mm.


**Figure B.1**: NET iCube NS4133BU

**Table B.1**: NET iCube NS4133BU Specifications

| | |
|---|---|
| Sensor Resolution | 1280 × 1024 pixels |
| Sensor Size | 1/1.8″ |
| Pixel Size | 5.3µm × 5.3µm |
| Lens Mount | C/CS-mount |
| Full Resolution Frame Rate | 25 fps |
| Interface | USB 2.0 |

### B.1.2 Prosilica EC-1350

The Prosilica EC-1350 is a 1.4 megapixel CCD camera based on the IIDC/DCAM specification.

**Figure B.2:** Prosilica EC-1350

**Table B.2:** Prosilica EC-1350 Specifications

| | |
|---|---|
| Sensor Resolution | 1360 × 1024 pixels |
| Sensor Size | 1/2″ |
| Pixel Size | 4.65µm × 4.65µm |
| Lens Mount | C-mount |
| Full Resolution Frame Rate | 18.5 fps |
| Interface | IEEE 1394A (FireWire) |

## B.2 Lenses

The internal parameters that describe the camera are primarily affected by the selection of lenses. The models used at some stage during the experiments presented in this dissertation are shown here with specifications.

### B.2.1 NET SV-0813V

The NET SV-0813V is a small form factor lens with low distortion and adjustable subject distance.

**Figure B.3:** NET SV-0813V

### B.2.2 Computar H10Z1218-MP

The Computar H10Z1218-MP is a large lens with adjustable and motorized zoom and focus settings.

**Table B.3**: NET SV-0813V Specifications

| | |
|---|---|
| Focal Length | 7mm - 8mm |
| Max. Aperture Ratio | 1 : 1.3 |
| Max. Sensor Size | 2/3″ |
| Mount Type | C-mount |


**Figure B.4**: Computar H10Z1218-MP

**Table B.4**: Computar H10Z1218-MP Specifications

| | |
|---|---|
| Focal Length | 12mm - 120mm |
| Max. Aperture Ratio | 1 : 1.8 |
| Max. Sensor Size | 6.4mm × 4.8mm |
| Mount Type | C-mount |

### B.2.3 Computar M3Z1228C-MP

The Computar M3Z1228C-MP is a medium size lens with low distortion and adjustable, but manual, focus and zoom settings.


**Figure B.5**: Computar M3Z1228C-MP

**Table B.5**: Computar M3Z1228C-MP Specifications

| | |
|---|---|
| Focal Length | 12mm - 36mm |
| Max. Aperture Ratio | 1 : 2.8 |
| Max. Sensor Size | 8.8mm × 6.6mm |
| Mount Type | C-mount |

## B.3 Other

Deploying the various experimental apparatuses in this dissertation required other equipement aside from cameras and lenses. Such equipment is described here.

### B.3.1 Mitsubishi RV-1A

The Mitsubishi RV-1A is a robotic manipulator with six degrees of freedom. Communication with the robot is possible through the RS-232 serial port link. The robot can be queried for the joint configuration of the end effector, which can in turn be converted into a pose in $SE(3)$.

**Figure B.6**: Mitsubishi RV-1A

### B.3.2 General Purpose Computer

With the exception of camera calibration and other minor operations, the majority of the experiments and simulations in this dissertation were performed in a general purpose computer. The specifications of this computer are shown bellow:

**Figure B.7**: General Purpose Computer

**Table B.6**: General Purpose Computer Specifications

| | |
|---|---|
| Processor | Intel i7 2.00GHz |
| Main Memory | 8GB DDR3 |
| Hard Drive | 250GB SSD SATA |
| Graphics | Intel HD 4000 |

# APPENDIX C

# Written Permission

This appendix contains a copy of the email in which I obtained written permission to use the contents of my previous publications on my dissertation.

University of Windsor [redacted]

## Written Permission

**Jose Alarcon Herrera** < @uwindsor.ca> Tue, Sep 16, 2014 at 11:04 AM
To: Xiang Chen < @uwindsor.ca>

Dr. Chen,

The contents of the following publications appear in my Ph.D. Dissertation as follows:

[3] appears in section 2.2
[2] appears in chapter 3
[1] appears in chapters 4, 5, and 6.

In your capacity as the main copyright holder, Do I have your permission to use these materials in my Ph.D. Dissertation?

[1] J.L. Alarcon-Herrera, X. Chen, and X. Zhang, "A Tensor-Based Vision Distance for Optimization of Multi-Camera Systems," to be submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence, 2014.

[2] J.L. Alarcon-Herrera and X. Chen, "Graph-Based Deployment of Visual Sensor Networks and Optics Optimization," submitted to IEEE/ASME Transactions on Mechatronics, manuscript no. TMECH-07-2014-3852, 2014.

[3] A. Mavrinac, X. Chen, and J.L. Alarcon-Herrera, "Semi-Automatic Model-Based View Planning for Active Triangulation 3D Inspection Systems," to appear in IEEE/ASME Transactions on Mechatronics, manuscript no. TMECH-05-2012-2343.R3, 2014.

Thank you
--
Jose Alarcon
[redacted]

**Xiang Chen** < @uwindsor.ca> Tue, Sep 16, 2014 at 12:37 PM
To: Jose Alarcon Herrera < @uwindsor.ca>
Cc: Xiang Chen < @uwindsor.ca>

Yes, you can use materials in these papers.

Regards,

==========
Xiang Chen, Ph.D., P. Eng.
Professor and Acting Head
Dept of Electrical and Computer Engineering
[redacted]
==========

Figure C.1: Written Permission

# Glossary of Terms

Glossary of Terms

**aperture**

In an optical system, the opening through which light passes. 15, 21

**blur circle**

An optical spot caused by a cone of light rays from the lens not coming into perfect focus when imaging a point source. 13

**coverage function**

A function dependent on the sensor system, environment, and task mapping the stimulus space to a bounded numeric range. 12, 14, 19

**coverage model**

A set of parameters modeling the sensor system, environment, and task, along with a specification of their relations, defining a coverage function in terms of the parameters. 6, 9, 12, 13, 15, 19, 20

**depth of field**

A range along the focal axis of a camera in which objects appear acceptably in focus (i.e. points map to acceptably small blur circles), according to some criterion. 15

**deterministic occlusion**

A model of occlusion caused by static objects or, more generally, objects with known dynamics. 12

**field of view**

The quadrilateral pyramid enclosing the subspace of $\mathbb{R}^3$ within which points project onto the image sensor. 4, 10, 14, 15, 20, 22, 23

**focal length**

In the pinhole camera model, the distance from the focal point to the image plane along the optical axis. Refers to the effective focal length of a complex optical system. 10, 15, 21–23

**focus**

As applied to a digital camera and its optical system, the property of sharpness in the imaging of an object, quantified by the blur circle diameter; dependent on the depth of the object and optical properties. 5, 13, 15, 21

**image plane**

In the pinhole camera model, the plane on which the three-dimensional scene is projected through the camera aperture (optical center). Parallel to the $x$-$y$ plane (focal plane) in the camera coordinate system. 18, 21, 39

**occlusion**

The effect of an opaque object obstructing the line of sight to a point beyond the object from the viewpoint. 8, 10, 18, 21

**optical axis**

The imaginary line which defines the path along which light propagates through the lens system of a camera; the principal axis or principal ray. Coincides with the axis of rotational symmetry. Considered the positive $z$-axis of the camera coordinate system. 15, 18, 31, 36, 39–41

**overlap graph**

A graph encapsulating the pairwise coverage overlap topology of a set of cameras. 24, 25, 49

**pinhole camera model**

Mathematical description of the projection of a 3D point onto a 2D image plane in an ideal pinhole camera (point aperture, no lens effects). 12

**pose**

A rigid Euclidean transformation in $SE(n)$. 10, 11, 21, 35, 41

**relevance function**

A function mapping the task point set to a bounded numeric range based on relevance to the task. 12

**resolution**

The smallest change detectable by a sensor in the quantity that it measures. For digital cameras, usually measured in terms of units of length per pixel; dependent on the depth of the object being imaged and optical and sensor properties. 5, 9, 13–15, 18, 21–23

**self-occlusion**

The phenomenon whereby some part of a (complex) object interrupts the ray from a feature on some other part of its surface to a camera. 36

**subject distance**

In an optical system, the distance at which objects are projected onto the image plane in focus. 73

**task**

A process to be carried out by a sensor system online, one or more of which comprise the end objective of the system. 8–10, 12, 13, 16, 17, 19–21, 31–33, 44, 48, 49

**tensor**

A geometric object that describes a linear relation between vectors, scalars, and other tensors. 12, 32–36, 38, 45, 47

**view**

The set of configuration parameter values of a multi-camera system; the instance of a vision sensor system model (equivalent to a viewpoint for single-camera systems). 9, 11

**view angle**

The angle between the surface normal of a stimulus point located on a surface and the ray from the camera's principal point through the stimulus point. 5, 13, 14

**viewing frustum**

The pyramidal frustum of visual coverage obtained by truncating the field of view with depth limits imposed by resolution and/or focus constraints. 14, 15, 31, 32, 38, 44, 45, 48

**viewpoint**

The combined set of intrinsic and extrinsic parameter values of a camera; the instance of a vision sensor model. 8, 10, 12, 17–26

**vision graph**

A graph encapsulating the pairwise coverage strength topology of a set of cameras. 23

# Propositions and Definitions

## List of Propositions



## List of Definitions

# Bibliography


[1] A. Mavrinac and X. Chen, "Modeling Coverage in Camera Networks: A Survey," *International Journal on Computer Vision*, vol. 101, no. 1, pp. 205–226, 2013.

[2] J. Shi and C. Tomasi, "Good Features to Track," in *IEEE Computer Vision and Pattern Recognition*, 1994, pp. 593–600.

[3] K. Tarabanis, R. Tsai, and P. Allen, "Analytical Characterization of the Feature Detectability Constraints of Resolution, Focus, and Field-of-View for Vision Sensor Planning," *Image Understanding*, vol. 59, no. 3, pp. 340–358, 1994.

[4] A. Mavrinac, "Modeling and Optimizing the Coverage of Multi-Camera Systems," Ph.D. Thesis, University of Windsor, 2012.

[5] J. Balog, "Extreme Ice Survey," *AGU Fall Meeting Abstracts*, no. 12, 2008. [Online]. Available: http://adsabs.harvard.edu/abs/2008AGUFMGC51A0652B

[6] D. Casbeer, R. Beard, T. McLain, L. Sai-Ming, and R. Mehra, "Forest fire monitoring with multiple small UAVs," in *American Control Conference*, 2005, pp. 3530–3535.

[7] L. Merino, F. Caballero, J. Dios, and A. Ollero, "Cooperative Fire Detection using Unmanned Aerial Vehicles," in *IEEE International Conference on Robotics and Automation*, 2005, pp. 1884–1889.

[8] S. Formentin, D. Berretta, N. Urbano, I. Boniolo, P. De Filippi, and S. Savaresi, "A Parking Assistance System for Small-Scale Boats," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 6, pp. 1844–1849, 2013.

[9] G. Panahandeh and M. Jansson, "Vision-Aided Inertial Navigation Based on Ground Plane Feature Detection," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 4, pp. 1206–1215, 2014.

[10] E. Ul Haq, S. Pirzada, J. Piao, T. Yu, and S. Hyunchul, "Image processing and vision techniques for smart vehicles," in *IEEE International Symposium on Circuits and Systems*, 2012, pp. 1211–1214.

[11] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage Control for Mobile Sensing Networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[12] R. Olfati-Saber, "Flocking for Multi-Agent Dynamic Systems: Algorithms and Theory," *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, 2006.

[13] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and Cooperation in Networked Multi-Agent Systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[14] B. Lei, W. Li, and F. Zhang, "Flocking Algorithm for Multi-Robots Formation Control with a Target Steering Agent," in *IEEE International Conference on Systems, Man and Cybernetics*, 2008, pp. 3536–3541.

[15] W. R. Scott, "Model-Based View Planning," *Machine Vision and Applications*, vol. 20, no. 1, pp. 47–69, 2009.

[16] F. Qureshi and D. Terzopoulos, "Surveillance camera scheduling: a virtual vision approach," *Multimedia Systems*, vol. 12, no. 3, pp. 269–283, 2006.

[17] W. Hu, T. Tan, L. Wang, and S. Maybank, "A Survey on Visual Surveillance of Object Motion and Behaviors," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 34, no. 3, pp. 334–352, 2004.

[18] V. Lepetit and F. Pascal, "Monocular Model-Based 3D Tracking of Rigid Objects: A Survey," *Foundations and Trends in Computer Graphics and Vision*, vol. 1, no. 1, pp. 1–89, 2011.

[19] A. Cavoukian, "Guidelines for the Use of Video Surveillance Cameras in Public Places," Information and Privacy Commissioner of Ontario, Tech. Rep., 2007. [Online]. Available: www.ipc.on.ca/images/Resources/video-e.pdf

[20] J. Alarcon-Herrera and X. Chen, "Online Configuration of PTZ Camera Networks," in *ACM/IEEE International Conference on Distributed Smart Cameras*, 2012, pp. 1–6.

[21] ——, "Consensus Algorithms in a Multi-Agent Framework to Solve PTZ Camera Reconfiguration in UAVs," in *International Conference on Intelligent Robotics and Applications*, 2012, pp. 1–6.

[22] A. Mavrinac and X. Chen, "Optimizing Load Distribution in Camera Networks with a Hypergraph Model of Camera Topology," in *ACM/IEEE International Conference Distributed Smart Cameras*, 2011.

[23] J. Alarcon-Herrera, X. Chen, and M. Ahmadi, "Re-Configuration Strategy for PTZ Camera Networks," in *FAC Symposium on Mechatronic Systems*, 2013, pp. 563–568.

[24] A. Mavrinac, X. Chen, and Y. Tan, "Coverage Quality and Smoothness Criteria for Real-Time View Selection in a Multi-Camera Network," *ACM Transactions on Sensor Networks*, vol. 10, no. 2, 2014.

[25] R. Pito, "A Solution to the Next Best View Problem for Automated Surface Acquisition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 10, pp. 1016–1030, 1999.

[26] C. Cowan and P. Kovesi, "Automatic Sensor Placement from Vision Task Requirements," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 3, pp. 407–416, 1988.

[27] K. Tarabanis and R. Y. Tsai, "Computing Viewpoints that Satisfy Optical Constraints," in *IEEE Conference on Computer Vision and Pattern Recognition*, 1991, pp. 152–158.

[28] Z. Cheng, D. Devarajan, and R. J. Radke, "Determining Vision Graphs for Distributed Camera Networks using Feature Digests," *EURASIP Journal of Applied Signal Processing*, 2007.

[29] M. A. Patricio, J. Carbó, O. Pérez, J. García, and J. M. Molina, "Multi-Agent Framework in Visual Sensor Networks," *EURASIP Journal on Advances in Signal Processing*, 2007.

[30] F. Castanedo, J. García, M. Patricio, and J. Molina, "Designing a Visual Sensor Network Using a Multi-Agent Architecture," in *International Conference on Practical Applications of Agents and Multi-Agent Systems*, 2009, pp. 430–439.

[31] K. Tarabanis, R. Tsai, and A. Kaul, "Computing occlusion-free viewpoints," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 3, pp. 279–292, 1996.

[32] F. Angella, L. Reithler, and F. Gallesio, "Optimal Deployment of Cameras for Video Surveillance Systems," in *IEEE Conference on Advanced Video and Signal Based Surveillance*, 2007, pp. 388–392.

[33] G. Olague and R. Mohr, "Optimal Camera Placement to Obtain Accurate 3D Point Positions," in *International Conference on Pattern Recognition*, no. 4, 1998, pp. 8–10.

[34] W. Zhong, J. Liu, M. Xue, and L. Jiao, "A Multiagent Genetic Algorithm for Global Numerical Optimization," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 34, no. 2, pp. 1128–1141, 2004.

[35] R. Malik and P. Bajcsy, "Automated Placement of Multiple Stereo Cameras," in *Workshop on Omnidirectional Vision, Camera Networks and Non-Classical Cameras*, 2008.

[36] C. Wang, F. Qi, and G. M. Shi, "Nodes Placement for Optimizing Coverage of Visual Sensor Networks," *Advances in Multimedia Information Processing*, pp. 1144–1149, 2009.

[37] Y. Jiang, J. Yang, W. Chen, and W. Wang, "A Coverage Enhancement Method of Directional Sensor Network Based on Genetic Algorithm for Occlusion-Free Surveillance," *IEEE International Conference on Computational Aspects of Social Networks*, pp. 311–314, 2010.

[38] K. K. Reddy and N. Conci, "Camera positioning for global and local coverage optimization." in *ACM/IEEE International Conference on Distributed Smart Cameras*, 2012, pp. 1–6.

[39] A. Mavrinac, X. Chen, and J. Alarcon-Herrera, "Semiautomatic Model-Based View Planning for Active Triangulation 3-D Inspection Systems," *IEEE/ASME Transactions on Mechatronics*, 2014.

[40] R. Bodor, A. Drenner, M. Janssen, P. Schrater, and N. Papanikolopoulos, "Mobile Camera Positioning to Optimize the Observability of Human Activity Recognition Tasks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 1564–1569.

[41] R. Bodor, A. Drenner, P. Schrater, and N. Papanikolopoulos, "Optimal Camera Placement for Automated Surveillance Tasks," *Journal of Intelligent and Robotic Systems*, vol. 50, no. 3, pp. 257–295, 2007.

[42] H. González-Banos and J. C. Latombe, "A Randomized Art-Gallery Algorithm for Sensor Placement," in *Annual Symposium Computational Geometry*, 2001, pp. 232–240.

[43] J. O'rourke, *Art gallery theorems and algorithms*. Oxford University Press Oxford, 1987, vol. 57.

[44] M. Marengoni, B. A. Draper, A. Hanson, and R. Sitaraman, "A System to Place Observers on a Polyhedral Terrain in Polynomial Time," *Image and Vision Computing*, vol. 18, no. 10, pp. 773–780, 1997.

[45] A. Bottino and A. Laurentini, "A Practical Iterative Algorithm for Sensor Positioning," in *IEEE International Conference on Emerging Technologies and Factory Automation*, vol. 1, 2005, pp. 1089–1092.

[46] U. M. Erdem and S. Sclaroff, "Automated Camera Layout to Satisfy Task-Specific and Floor Plan-Specific Coverage Requirements," *Computer Vision and Image Understanding*, vol. 103, no. 3, pp. 156–169, 2006.

[47] A. Mittal and L. S. Davis, "Visibility Analysis and Sensor Planning in Dynamic Environments," in *European Conference on Computer Vision*, 2004.

[48] ——, "A General Method for Sensor Planning in Multi-Sensor Systems: Extension to Random Occlusion," *International Journal of Computer Vision*, vol. 76, no. 1, pp. 31–52, 2008.

[49] E. Hörster and R. Lienhart, "Calibrating and Optimizing Poses of Visual Sensors in Distributed Platforms," *Multimedia Systems*, vol. 12, no. 3, pp. 195–210, 2006.

[50] X. Chen and J. Davis, "An Occlusion Metric for Selecting Robust Camera Configurations," *Machine Vision and Applications*, vol. 19, no. 4, pp. 217–222, 2007.

[51] A. Kansal, W. Kaiser, G. Pottie, M. Srivastava, and G. Sukhatme, "Reconfiguration methods for mobile sensor networks," *ACM Transactions on Sensor Networks*, vol. 3, no. 4, pp. 22:1–22:28, 2007.

[52] J. Zhao, S. C. Cheung, and T. Nguyen, "Optimal Camera Network Configurations for Visual Tagging," *IEEE Journal of Selected Topics in Signal Processing.*, vol. 2, no. 4, pp. 464–479, 2008.

[53] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg, "Robust sensor placements at informative and communication-efficient locations," *ACM Transactions on Sensor Networks*, vol. 7, no. 4, pp. 31:1–31:33, 2011.

[54] S. Ram, K. R. Ramakrishnan, P. K. Atrey, V. K. Singh, and M. S. Kankanhalli, "A Design Methodology for Selection and Placement of Sensors in Multimedia Surveillance Systems," in *ACM International Workshop on Video Surveillance and Sensor Networks*, 2006, pp. 121–130.

[55] M. D. Mackay, R. G. Fenton, and B. Benhabib, "Multi-Camera Active Surveillance of an Articulated Human Form – An Implementation Strategy," *Computer Vision and Image Understanding*, vol. 115, no. 10, pp. 1395–1413, 2011.

[56] L. Fiore, G. Somasundaram, A. Drenner, and N. Papanikolopoulos, "Optimal Camera Placement with Adaptation to Dynamic Scenes," in *IEEE International Conference on Robotics and Automation*, 2008, pp. 956–961.

[57] C. Piciarelli, C. Micheloni, and G. L. Foresti, "PTZ Camera Network Reconfiguration," in *ACM/IEEE International Conference on Distributed Smart Cameras*, 2009, pp. 1–7.

[58] ——, "Occlusion-Aware Multiple Camera Reconfiguration," in *ACM/IEEE International Conference on Distributed Smart Cameras*, 2010, pp. 88–94.

[59] C. Piciarelli, C. Micheloni, and G. Foresti, "Automatic Reconfiguration of Video Sensor Networks for Optimal 3D Coverage," in *ACM/IEEE International Conference on Distributed Smart Cameras*, 2011, pp. 1–6.

[60] F. Z. Qureshi and D. Terzopoulos, "Surveillance in Virtual Reality: System Design and Multi-Camera Control," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[61] ——, "Planning Ahead for PTZ Camera Assignment and Handoff," in *ACM/IEEE International Conference on Distributed Smart Cameras*, 2009, pp. 1–8.

[62] W. Starzyk and F. Qureshi, "Learning Proactive Control Strategies for PTZ Cameras," in *ACM/IEEE International Conference on Distributed Smart Cameras*, 2011, pp. 1–6.

[63] K. Y. Chow, K. S. Lui, and E. Y. Lam, "Maximizing Angle Coverage in Visual Sensor Networks," in *IEEE International Conference on Communications*, 2007, pp. 3516–3521.

[64] ——, "Achieving 360 ° Angle Coverage with Minimum Transmission Cost in Visual Sensor Networks," in *IEEE Wireless Communications and Networking Conference*, 2007, pp. 4112–4116.

[65] ——, "Wireless Sensor Networks Scheduling for Full Angle Coverage," *Multidimensional Systems and Signal Processing*, vol. 20, no. 2, pp. 101–119, 2008.

[66] X. Yang, L. Prasad, and L. Latecki, "Affinity Learning with Diffusion on Tensor Product Graph," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 28–38, 2013.

[67] R. Sivalingam, D. Boley, V. Morellas, and N. Papanikolopoulos, "Tensor Sparse Coding for Positive Definite Matrices," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 3, pp. 592–605, 2014.

[68] T. Tung and T. Matsuyama, "Geodesic Mapping for Dynamic Surface Alignment," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 5, pp. 901–913, 2014.

[69] R. Tron, R. Vidal, and A. Terzis, "Distributed Pose Averaging in Camera Networks via Consensus on SE(3)," in *ACM/IEEE International Conference on Distributed Smart Cameras*, no. 3, 2008.

[70] C. Soto, S. Bi, and A. Roy-Chowdhury, "Distributed Multi-Target Tracking in a Self-Configuring Camera Network," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 1486–1493.

[71] S. Bi, A. T. Kamal, C. Soto, D. Chong, J. A. Farrell, and A. K. Roy-Chowdhury, "Tracking and Activity Recognition Through Consensus in Distributed Camera Networks," *IEEE Transactions on Image Processing*, vol. 19, no. 10, pp. 2564–2579, 2010.

[72] P. Basser and C. Pierpaoli, "Microstructural and Physiological Features of Tissues Elucidated by Quantitative-Diffusion-Tensor MRI," *Journal of Magnetic Resonance*, vol. 111, no. 1, pp. 209–219, 1996.

[73] D. Le Bihan, J. Mangin, C. Poupon, C. Clark, S. Pappata, N. Molko, and H. Chabriat, "Diffusion Tensor Imaging: Concepts and Applications," *Journal of Magnetic Resonance Imaging*, vol. 13, pp. 534–546, 2001.

[74] Z. Wang and B. Vemuri, "DTI Segmentation Using an Information Theoretic Tensor Dissimilarity Measure," *IEEE Transactions on Medical Imaging*, vol. 24, no. 10, pp. 1267–1277, 2005.

[75] Y. Liu and K. Chan, "Tensor Distance Based Multilinear Locality-Preserved Maximum Information Embedding," *IEEE Transactions on Neural Networks*, vol. 21, no. 11, pp. 1848–1854, 2010.

[76] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*. SpringerVerlag, 2004.

[77] E. Hecht, *Optics*, 2nd ed. Addison Wesley, 1987.

[78] A. Mavrinac, J. Alarcon-Herrera, and X. Chen, "A Fuzzy Model for Coverage Evaluation of Cameras and Multi-Camera Networks," in *ACM/IEEE International Conference on Distributed Smart Cameras*, 2010, pp. 95–102.

[79] J. Alarcon-Herrera, A. Mavrinac, and X. Chen, "Sensor Planning for Range Cameras via a Coverage Strength Model," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2011, pp. 838–843.

[80] G. Tarbox and S. Gottschlich, "Planning for Complete Sensor Coverage in Inspection," *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 84–111, 1995.

[81] Y. Chen, S. Shih, Y. Hung, and C. Fuh, "Simple and Efficient Method of Calibrating a Motorized Zoom Lens," *Image and Vision Computing*, vol. 19, no. 14, pp. 1099–1110, 2001.

[82] M. Sarkis, C. T. Senft, and K. Diepold, "Modeling the Variation of the Intrinsic Parameters of an Automatic Zoom Camera System using Moving Least-Squares," in *IEEE International Conference on Automation Science and Engineering*, 2007, pp. 560–565.

[83] ——, "Partitioned Moving Least-Squares Modeling of an Automatic Zoom Lens Camera," in *International Conference on Control, Automation and Systems*, 2007, pp. 544–547.

[84] J. Alarcon-Herrera, X. Chen, and X. Zhang, "Viewpoint Selection for Vision Systems in Industrial Inspection," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 4934–4939.

[85] J. Alarcon-Herrera and X. Chen, "Deployment of Visual Sensor Networks Using a Graph-Based Approach," in *IEEE International Symposium on Circuits and Systems*, 2014, pp. 2575–2578.

[86] G. Strang, *Introduction to linear algebra.* Cambridge, 2003.

[87] G. Murphy, *C*-algebras and operator theory.* Academic press San Diego, 1990, vol. 288.

[88] A. Naylor and G. Sell, *Linear Operator Theory in Engineering and Science*, ser. Applied Mathematical Sciences. Springer, 2000, vol. 40.

[89] A. Mavrinac and J. Alarcon-Herrera, "Adolphus." [Online]. Available: http://github.com/ezod/adolphus.

[90] Y. Ma, J. Kosecka, and S. Sastry, "Optimization Criteria and Geometric Algorithms for Motion and Structure Estimation," *International Journal of Computer Vision*, vol. 44, no. 3, pp. 219–249, 2001.

[91] S. Sra, S. Nowozin, and S. Wright, *Optimization for Machine Learning.* MIT Press, 2011.

[92] R. Murray, Z. Li, and S. Sastry, *A Mathematical Introduction to Robotic Manipulation.* CRC Press, 1994.

[93] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *IEEE Conference on Neural Networks*, 1995, pp. 1942–1948.

[94] ——, *Swarm Intelligence.* Morgan Kaufmann, 2001.

[95] T. Moller and B. Trumbore, "Fast, Minimum Storage Ray/Triangle Intersection," *Journal of Graphic Tools*, vol. 2, no. 1, pp. 21–28, 1997.

[96] M. Floater and K. Hormann, "Parameterization of Triangulations and Unorganized Points," in *Tutorials on Multiresolution in Geometric Modelling.* Springer Berlin Heidelberg, 2002, pp. 287–316.

[97] ——, "Surface Parameterization: a Tutorial and Survey," in *Advances in Multiresolution for Geometric Modelling.* Springer Berlin Heidelberg, 2005, pp. 157–186.

[98] L. Wang, B. Yuan, and Z. Miao, "3D Point Clouds Parameterization Alogrithm," in *International Conference on Signal Processing*, 2008, pp. 1410–1413.

[99] M. Zwicker and C. Gotsman, "Meshing Point Clouds Using Spherical Parameterization," in *Eurographics Symposium on Point-Based Graphics*, 2004, pp. 1–8.

[100] G. Tewari, C. Gotsman, and S. J. Gortler, "Meshing Genus-1 Point Clouds using Discrete One-Forms," *Computers and Graphics*, vol. 30, no. 12, pp. 917–926, 2006.

# Vita Auctoris

Jose Alarcon was born in 1986 in Durango, Mexico. He received the degree of Bachelor of Applied Science in Electrical Engineering from the Institute of Technology of Durango in 2009. After working in the automotive industry he moved to Canada to pursue graduate studies at the University of Windsor working on visual-based motion control in 2010. After eight months in the program of Master of Applied Science he was transfered to the program of Doctor of Philosophy under the supervision of Dr. Xiang Chen. His further work in the area of computer vision culminated in this doctoral dissertation in 2014. Jose is expected to obtain the degree of Doctor of Philosophy in Electrical Engineering in October of 2014.